Hélio Ishii Stroparo

Rafael Ávila Leite

GERÊNCIA DE REDES LAN

Monografia apresentada à disciplina Projeto de Graduação do Curso de Tecnologia em Informática da Faculdade de Ciências Exatas e de Tecnologia da Universidade Tuiuti do Paraná como requisito para obtenção do grau de Tecnólogo em Informática.

Orientador: Roberto Néia Amaral

CURITIBA

2006

Agradecemos aos professores do Curso de Tecnologia em Informática pelo incentivo constante no decorrer de todo esse tempo de estudo.

Dedicamos esta obra a nossas famílias, primeiramente, pelo apoio, e em especial ao meu orientador professor Roberto Néia Amaral, por nos abrir o campo de visão para novos horizontes.

Educação é aquilo que resta quando nos esquecemos daquilo que nos foi ensinado.

Michael Hammer

RESUMO

Este trabalho tem por objetivo o desenvolvimento de um protótipo gratuito, onde seja possível gerenciar os ativos de uma rede de dados. Busca-se realizar a proposta, com o intuito de proporcionar aos administradores de redes, através de um algoritmo estruturado nos conceitos de gerenciamento de redes, redes de dados e linguagens de programação, melhora na disponibilidade dos recursos de redes bem como na performance de atendimento aos usuários. O protótipo propõe um gerenciamento de redes LAN que permita ao administrador de redes tomar decisões quanto aos problemas que estão ocorrendo, como também os que podem ocorrer, favorecendo a análise crítica do funcionamento de uma rede de dados.

Palavras-chave: Rede de Dados, Administradores de Redes, Gerenciamento.

LISTA DE FIGURAS

FIGURA 1 – Repetidor 16
FIGURA 2 – *Hubs* 16
FIGURA 3 – *Bridges* 17
FIGURA 4 – *Switchs* 18
FIGURA 5 – Roteadores 19
FIGURA 6 – Estrutura Geral do Protótipo 39

LISTA DE TABELAS

QUADRO 1 – Comparação entre topologias 23

SUMÁRIO

1 INTRODUÇÃO 10
2 REFERÊNCIAL TEÓRICO 12
2.1 TIPOS DE REDES DE DADOS 13
2.1.1 LAN – *Local Area Network* 13
2.1.2 WAN – *Wide Area Network* 15
2.2 TIPOS DE DISPOSITIVOS DE REDE DE DADOS 15
2.2.1 Repetidor 15
2.2.2 *Hub* 16
2.2.3 *Bridges* 17
2.2.4 *Switch* 18
2.2.5 Roteador 19
2.3 TIPOS DE TOPOLOGIAS DE REDE DE DADOS 19
2.3.1 Estrela 20
2.3.2 Anel 21
2.3.3 Barramento 22
2.4 TIPOS DE PADRÕES DE TRANSMISSÃO EM REDE DE DADOS 23
2.4.1 *Ethernet* 23
2.4.2 *Token Ring* 24
2.5 GERENCIAMENTO DE REDES 25
2.5.1 SNMP 25
2.5.2 RMON 26
2.5.3 Sistemas de Gerenciamento OSI 27
2.6 NECESSIDADES DO GERENCIAMENTO DE REDES 29
2.7 HISTÓRICO DO GERENCIAMENTO DE REDES 31
2.8 SOFTWARE DE GERENCIAMENTO DE REDES 36
2.8.1 HP Open View 36
2.8.2 NetView 36
2.8.3 Spectrum 37
2.8.4 SunNet Manager 37
3 METODOLGIA 38
3.1 LINGUAGEM DE PROGRAMAÇAO 38
3.2 PRÉ-REQUISITOS 39
3.3 ESTRUTURA DE FUNCIONAMENTO DO PROTÓTIPO 39
3.4 PRINCIPIO DE FUNCIONAMENTO DO PROTÓTIPO 40
3.5 MANUAL DO USUÁRIO 40
3.6 MANUAL DE INSTALAÇÃO 48
3.7 DFD – DIAGRAMA DE FLUXO DE DADOS 63

3.8 DER – DIAGRAMA DE ENTIDADE E RELACIONAMENTO 63
4 TESTES 64
5 CONLUSÃO 65
REFERÊNCIAS 67
ANEXO 1 – BIBLIOTECA PING 68
ANEXO 2 – BIBLIOTECA ICMP 78
APÊNDICE 1 – Form_Banco 86
APÊNDICE 2 – Form_CadastroComputadores 87
APÊNDICE 3 – Form_CadGrupos 89
APÊNDICE 4 – Form_GeraRelatorios 91
APÊNDICE 5 – Form_RelatorioAtivo 92
APÊNDICE 6 – Form_RelatoriosPeriodo 94
APÊNDICE 7 – Biblioteca Ping Modificada 97

1 INTRODUÇÃO

A área de gerenciamento de redes foi inicialmente impulsionada pela necessidade de monitoração e controle do universo de ativos que compõem as redes de dados.

Atualmente as redes de computadores e os seus recursos associados, além das aplicações distribuídas, tem se tornado fundamental e de tal importância para uma organização, onde o tempo de indisponibilidade afeta diretamente o desempenho das organizações.

Isto significa que o nível de falhas e de indisponibilidade considerados aceitáveis estão diminuindo, podendo chegar a até zero, dependendo da importância da rede para uma organização.

O objetivo geral deste projeto é desenvolver um protótipo que permita gerenciar o estado dos ativos de uma rede de dados. O protótipo será desenvolvido em Delphi para ser utilizado com o sistema operacional *Windows 2000 ou XP*. Os dados coletados durante o gerenciamento serão disponibilizados em banco de dados *Interbase*, possibilitando que os administradores de redes reconheçam os pontos vulneráveis de suas redes e planejar futuros investimentos melhorias em sua disponibilidade.

Em um primeiro momento foram realizados estudos com o objetivo de adquirir conhecimento sobre os programas e softwares já existentes para gerenciamento. Nesse estudo não se trata de validar ou invalidar as rotinas de gerenciamento, mas de compreendê-los e de saber distinguir as implicações de cada um e os métodos que podem ser utilizados para implementá-los.

Na seqüência do trabalho foi realizada uma revisão bibliográfica com o objetivo de fundamentar o estado do conhecimento neste presente, com o intuito de se conhecer o que já foi realizado em termos de pesquisa sobre o assunto. Esta fundamentação teórica foi direcionada ao foco principal deste trabalho, o qual é o gerenciamento de redes LAN.

Após a pesquisa para verificar as principais idéias que fundamentaram o tema foi apresentada a proposta que consiste em uma alternativa para o desenvolvimento de um protótipo para o gerenciamento de redes LAN.

2 REFERENCIAL TEÓRICO

Computadores interconectados podem trabalhar mais pelos usuários, e quando as pessoas trabalham em equipes, concretizam tarefas inteiras, num menor espaço de tempo e com menos esforço. Pode-se imaginar uma rede como um recurso valioso, projetada para apoiar uma equipe de usuários.

Interconectar os computadores, assim como gerenciar um grupo de pessoas é sem dúvida um desafio. O custo pode ter uma enorme variação. Os benefícios de se conectar os recursos podem ser grandes e podem significar um avanço incalculável de benefícios que um micro isolado nunca poderia apresentar.

Atenta aos possíveis benefícios e recompensas, e apesar dos riscos, as empresas estão interconectando seus computadores em rítimo acelerado.

Antigamente as redes eram de difícil instalação e manutenção exigindo mão de obra altamente qualificada, mas atualmente isto está mudando, hoje encontra-se componentes de hardware (*kits*) para instalação de redes que qualquer pessoa pode instalar.

Em um ambiente profissional é muito importante um responsável pelo bom funcionamento da rede, dentre as responsabilidades deste cita-se: coordenar tarefas, gerenciar problemas, monitorar processos, administrar usuários, entre outros.

Sem dúvida alguma, um dos maiores benefícios de uma rede é o compartilhamento de informações entre os usuários ou mesmo oferecer um meio de armazenamento final superior ao que é utilizado sem a rede.

2.1 TIPOS DE REDES DE DADOS

Atualmente pode-se contar com alguns tipos de rede quando a sua disposição física.

2.1.1 LAN – *Local Area Network* - Rede de alcance local

Redes locais (*LANs*) são basicamente um grupo de computadores interconectados e opcionalmente conectado a um servidor. Os usuários executam tarefas a partir de seus computadores. Entre as tarefas pode-se destacar os bancos de dados, planilhas e editores de texto. Normalmente tem-se um grupo destes usuários executando uma operação no servidor.

Atualmente encontram-se velocidades de comunicação das redes locais entre 10 e 100 Mbps (Mega bits por segundo) e 1 Gbps (Giga bits por segundo). A utilização de um padrão mais veloz estará na necessidade do usuário. Redes de 100 Mbps e 1 Gbps exigem hardware especial (placas, *switchs* e *hubs*).

Redes de 100 Mbps e 1 Gbps são recomendadas onde a velocidade é fundamental ao bom funcionamento, normalmente é utilizada em locais onde seja necessário troca de informações como som e imagem ou também em redes maiores.

As redes de 10 Mbps são instaladas em ambiente onde não é necessário executar aplicações no servidor com freqüência, pois isto tornaria lenta a velocidade de comunicação entre os equipamentos.

UNIVERSIDADE FUMEC
BIBLIOTECA
Sydney Antonio Rangel Santos
Barigui

Quanto ao tipo de cabeamento, para as redes de 10 Mbps pode-se utilizar tanto o cabo coaxial (10Base2) ou par trançado (10BaseT). Em uma rede 100 Mb ou 1 Gbps necessariamente utiliza-se cabos de par trançado (100BaseT e 1000BaseT).

Os módulos mais importantes de uma rede local são:

Servidores - São computadores que elevam a capacidade do processamento, cuja função é disponibilizar serviços à rede. Em geral essa máquina processa grandes volumes de dados requerendo por tanto *CPUs* (*Center Process Units*) rápidas e dispositivos de armazenamento de alta capacidade e acesso rápido. Estas máquinas poderão ser fornecidas por fabricantes especializados (IBM®, INTEL®, entre outros) e por serem máquinas especiais entre as outras, possuem características não encontradas nos modelos mais simples.

Em uma rede baseada em servidor, tem-se normalmente sistemas operacionais mais potentes, sendo necessário um estudo mais criterioso para a definição de qual Sistema Operacional utilizar.

PC *Desktop (Personal Computer)* - Os PCs clientes também conhecidos por *Workstation* (estação de trabalho) individuais de trabalho. A partir dela os usuários acessam informações no servidor (Banco de dados, entre outros) e rodam aplicações locais (Microsoft Word, Microsoft Excel, entre outros). O *hardware* da *workstation* podem variar, pois dependerá das informações a serem processados.

Recursos - Entre os recursos a serem utilizados na comunicação entre os equipamentos pode-se citar: os *Switchs* (comutadores), os *Hubs*, cabo coaxial, cabos par-trançado, placas de rede, repetidores, *Bridges* (pontes), roteadores, entre outros.

2.1.2 WAN – *Wide Area Network* - Rede de alcance remoto

As WANs utilizam linhas de transmissão oferecidas por operadoras de telecomunicações, com o objetivo de Interligação de computadores geograficamente distantes. A necessidade de transmissão de dados entre computadores surgiu com os *mainframes*, bem antes do aparecimento dos *PCs*. Com os *PCs* houve um aumento da demanda por transmissão de dados a longa distância. Isto levou ao surgimento de diversos serviços de transmissão de dados (RENPAC, TRANSDATA, MINASPAC). Os serviços são geralmente de aluguel de linhas privadas ou discadas permitindo a utilização de diversos protocolos tais como *SNA, PPP e TCP*, entre outros.

Hoje em dia as redes WANs utilizam meios de transmissão de fibras ópticas e protocolos de comunicação como o ATM (*Asynchronous Transfer Mode*) que permitem velocidades acima de 155 Mbps A velocidade passa a ser determinada pelos equipamentos que processam as informações (Clientes/Servidores) e não do meio físico.

2.2 TIPOS DE DISPOSITIVOS DE REDES DE DADOS

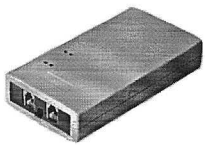
As redes de dados podem ser compostas por diversos dispositivos de *hardware*.

2.2.1 Repetidores

São equipamentos utilizados quando se deseja repetir o sinal enviado por um equipamento quando a distância a ser percorrida é maior do que o recomendado

(Exemplo para cabos UTP 100 metros). Ele realiza uma ampliação no sinal já fraco dando nova força para que chegue ao ponto de destino.

FIGURA 1: REPETIDOR - 2006

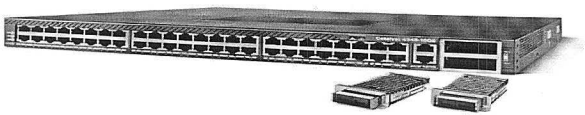


FONTE: Projeto de Redes

2.2.2 *Hubs*

São dispositivos utilizados para conectar os equipamentos que compõem uma LAN. Com o *Hub*, as conexões da rede são concentradas (por isto também chamado concentrador). O gerenciamento da rede é favorecido e a solução de problemas facilitada, uma vez que o defeito fica isolado no segmento de rede. Cada *hub* pode receber vários ativos de rede, atualmente existem *hubs* entre 4 e 48 portas (pode-se fazer a conexão entre *hubs* aumentando a capacidade final).

FIGURA 2: *HUB* - 2006



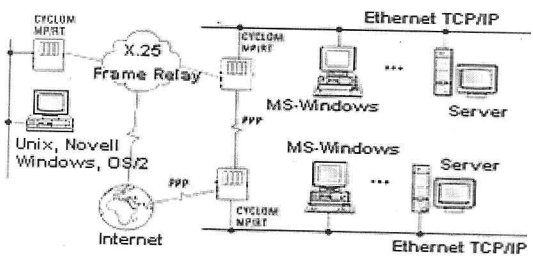
FONTE: CISCO SYSTEMS, Inc

2.2.3 *Bridges* (Pontes)

Seu trabalho é evitar o número excessivo de colisões quando um número grande de computadores compartilham o mesmo cabo. O *bridge* analisa os endereços dos pacotes de dados que chegam em cada porta, e transmitem o pacote para a outra porta somente quando concluem que o destino está neste outra porta.

O *Bridge* é uma ponte entre duas redes separadas, que passam a operar como uma única rede. Por exemplo, tem-se LAN da diretoria com a LAN da produção, cada uma com uma rede local, pode-se utilizar *bridges* para ligar as duas redes. Se for inviável ligar as duas redes através de cabos de rede normais, pode-se usar *bridges wireless* (sem fios). O tráfego é estabelecido entre as duas redes, que passam então a formar uma única rede.

FIGURA 3: EXEMPLODE UTILIZAÇÃO DE *BRIDGES* - 1995

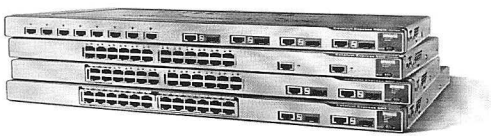


FONTE: Topologia de Redes

2.2.4 *Switchs*

Os *switchs* servem para interligar vários computadores e equipamentos de uma rede, possuem maior desempenho que o *hub* porque estabelece dinamicamente ligações entre as portas, estabelecendo canais independentes que podem operar de forma simultânea. Cada computador tem a rede à disposição até 100% do tempo. O *switch* opera de forma mais inteligente que o *hub*. Ele analisa os pacotes de dados que chegam a ele e descobre os endereços de origem e destino. A partir daí, enviará este pacote apenas para a porta correta. O resultado é que em um dado instante podem existir várias conexões internas simultâneas. Este chaveamento inteligente de conexões contribui para aumentar o desempenho total da rede e evitando colisões de dados.

FIGURA 4: Exemplo de *Switchs* - 2006

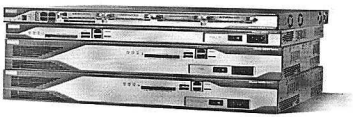


FONTE: CISCO SYSTEMS, Inc

2.2.5 Roteadores

Fazem o papel de guarda de trânsito, garantindo que os pacotes de mensagens sejam dirigidos a endereços certos na rede. O roteador faz a conexão entre duas redes diferentes, que podem inclusive ter características físicas e protocolos diferentes. Uma aplicação muito comum atualmente é a ligação entre uma rede local e a Internet.

FIGURA 5: Exemplo de Roteadores - 2006



FONTE: CISCO SYSTEMS, Inc

2.3 TIPOS DE TOPOLOGIAS DE REDES DE DADOS

A topologia de uma rede depende do projeto das operações, da confiabilidade e do seu custo operacional. Ao planejar uma rede, muitos fatores devem ser considerados, mas o tipo de participação dos nós é um dos mais importantes.

A topologia da rede é um nome dado ao arranjo dos cabos usados para interconectar os clientes e servidores. A maneira como eles são interligados tem

algumas implicações sobre a maneira como o sistema operacional de rede gerencia tanto os clientes quanto o fluxo de informações sobre a rede, as topologias mais comuns são estrela, anel, e barramento.

2.3.1 Estrela

Neste tipo de rede, todos os usuários comunicam-se com um nó central. Através do nó central, os usuários podem comunicar-se entre si e com processadores remotos ou terminais. O arranjo em estrela é a melhor escolha se o padrão de comunicação da rede for de um conjunto de estações secundárias que comunicam-se com o nó central. As situações onde isto mais acontece são aquelas em que o nó central está restrito às funções de gerente das comunicações e a operações de diagnósticos.

O nó central pode realizar outras funções além das de chaveamento e processamento normal. Por exemplo:

1. Compatibilizar a velocidade de comunicação entre o transmissor e o receptor.
2. Se o protocolo dos dispositivos, fonte e destino utilizarem diferentes protocolos, o nó central pode atuar como um conversor, permitindo duas redes de fabricantes diferentes se comunicar.

No caso de ocorrer falha em uma estação ou no elo de ligação com o nó central, apenas esta estação fica fora de operação. Entretanto, se uma falha ocorrer no nó central, todo o sistema pode ficar fora do ar. A solução deste problema seria a redundância, mas isto acarreta um aumento considerável dos custos.

A expansão de uma rede deste tipo de rede só pode ser feita até um certo limite, imposto pelo nó central: em termos de capacidade de chaveamento, número de circuitos concorrentes que podem ser gerenciados e número de nós que podem ser servidos.

O desempenho obtido numa rede em estrela depende da quantidade de tempo requerido pelo nó central para processar e encaminhar mensagens, e da carga de tráfego de conexão, ou seja, é limitado pela capacidade de processamento do nó central.

2.3.2 Anel

Uma rede em anel consiste de estações conectadas através de um caminho fechado. Nesta configuração, muitas estações remotas ao anel não se comunicam diretamente com o computador central.

Redes em anel são capazes de transmitir e receber dados em qualquer direção, mas as configurações mais usuais são unidirecionais, de forma a tornar menos sofisticados os protocolos de comunicação que asseguram a entrega da mensagem corretamente e em seqüência ao destino.

Quando uma mensagem é enviada por um nó, ela entra no anel e circula até ser retirada pelo nó destino, ou então até voltar ao nó fonte, dependendo do protocolo empregado. O último procedimento é mais desejável porque permite o envio simultâneo de um pacote para múltiplas estações. Outra vantagem é a de permitir á determinadas estações receber pacotes enviados por qualquer outra estação da rede, independentemente de qual seja o nó destino.

UNIVERSIDADE TUIUTI
BIBLIOTECA
Ayrton Zaniolo Rangel Santos
Bariguí

Os maiores problemas desta topologia são relativos a sua pouca tolerância a falhas. Qualquer que seja o controle de acesso empregado, ele pode ser perdido por problemas de falha e pode ser difícil determinar com certeza se este controle foi perdido ou decidir qual nó deve recriá-lo. Erro de transmissão e processamento podem fazer com que uma mensagem continue eternamente a circular no anel. A utilização de uma estação monitora, contornar estes problemas. Outras funções desta estação seriam: iniciar o anel, enviar pacotes de teste e diagnóstico e outras tarefas de manutenção. A estação monitora pode ser dedicada ou uma outra que assuma em determinado tempo essas funções.

Esta configuração requer que cada nó seja capaz de remover seletivamente mensagens da rede ou passá-las adiante para o próximo nó. Nas redes unidirecionais, se uma linha entre dois nós cair, todo sistema sai do ar até que o problema seja resolvido. Se a rede for bidirecional, nenhum ativo de rede ficará inacessível, já que poderá ser atingido pelo outro lado.

2.3.3 Barramento

Nesta configuração todos os nós (estações) se ligam ao mesmo meio de transmissão. O barramento é geralmente compartilhado em tempo e freqüência, permitindo transmissão de informação.

Nas redes em barramento, cada nó conectado à barra pode ouvir todas as informações transmitidas. Esta característica facilita as aplicações com mensagens do tipo difusão (para múltiplas estações).

Existe uma variedade de mecanismos para o controle de acesso ao barramento pode ser centralizado ou descentralizado. A técnica adotada para

acesso à rede é a multiplexação no tempo. Em controle centralizado, o direito de acesso é determinado por uma estação especial da rede. Em um ambiente de controle descentralizado, a responsabilidade de acesso é distribuída entre todos os nós. Relógios de prevenção em cada transmissor devem detectar e desconectar o nó que falha no momento da transmissão.

O desempenho de um sistema em barramento comum é determinado pelo meio de transmissão, número de nós conectados, controle de acesso, tipo de tráfego entre outros fatores. O tempo de resposta pode ser altamente dependente do protocolo de acesso utilizado.

QUADRO 1: COMPARAÇÃOENTRE TOPOLOGIAS

Tipos de Topologias	**Ponto Positivos**	**Pontos Negativos**
Topologia Estrela	É mais tolerante a falhas Fácil de instalar usuários Monitoramento centralizado	Custo de Instalação maior.
Topologia Anel (*Token Ring*)	Razoavelmente fácil de instalar Requer menos cabos Desempenho uniforme	Se uma estação pára todas param Os problemas são difíceis de isolar.
Topologia Barramento	Simples e fácil de instalar Requer menos cabos Fácil de entender	A rede fica mais lenta em períodos de uso intenso. Os problemas são difíceis de isolar.

2.4 TIPOS DE PADRÕES DE TRANSMISSÃO EM REDE DE DADOS PARA REDES LOCAIS

2.4.1 *Ethernet*

A rede *Ethernet* é a mais conhecida dentre as atualmente utilizadas, e está no mercado há mais tempo do que as outras tecnologias de rede. A redução dos preços e uma relativa alta velocidade de transmissão de dados fomentaram a ampla

utilização da *Ethernet*, que pode ser utilizada com topologia barramento (Coaxial) ou Estrela (Par trançado).

Neste tipo de rede, cada máquina "escuta" o tráfego na rede e se não ouvir nada, eles transmitem as informações. Se dois clientes transmitirem informações ao mesmo tempo, eles são alertados sobre a colisão, param a transmissão e esperam um período aleatório para cada um antes de tentar novamente, este método é conhecido como *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD). Como exemplo, suponha-se se deseje armazenar uma planilha no HD (*Hard Disk*) de uma outra máquina. Em *Ethernet*, o primeiro passo que sua placa de rede faz é escutar o que está acontecendo no cabo para determinar se, no momento, há alguém utilizando o cabo para transmitir dados. Essa é a parte *carrier sense* (detecção de portadora) do CSMA/CD.

Se a rede estiver ocupada sua placa continua tentando até que ela esteja livre. Uma vez que detecte que não existem dados trafegando então ela envia a planilha para outra máquina. Em caso de colisão, os dados são perdidos e cada um dos envolvidos na colisão aguardam um período aleatório para retransmitir não ocorrendo perdas para o usuário.

Dependendo do tipo de hardware utilizado na rede, o número de colisões crescem a medida que o número de estações aumentam. Uma forma de evitar as colisões é a utilização de *switchs* na rede.

2.4.2 *Token Ring*

O método de acesso de *token ring* (passagem de permissão) utiliza um método circular para determinar qual estação tem permissão para transmitir. O *token*

ring opera em topologia em anel e garante que todas as estações da rede tenham chance de transmitir dados. Ele alcança esse objetivo utilizando um padrão especial de bit conhecido como *token* ou permissão.

Em uma rede *token ring*, seu computador pacientemente monitora a rede até que ele veja um padrão especial de bits denominado permissão. Ao ver a transmissão ele envia um pacote de dados. Este pacote de dados viaja pelo anel e o destinatário recebe na passagem. Quando o pacote retornar ao transmissor ele passa o *token* para a próxima estação. Este processo se repete infinitamente. Os tempos necessários são medidos em frações de segundos.

2.5 GERENCIAMENTO DE REDES

A área de gerência de redes, foi inicialmente impulsionada pela necessidade de monitoração e controle dos dispositivos que compõem as redes de comunicação. Com esta crescente necessidade de gerenciamento, fez-se necessário que padrões para ferramentas fossem estabelecidos.

2.5.1 SNMP

O protocolo SNMP (*Simple Network Management Protocol* – protocolo simples de gerenciamento de redes) refere-se a um conjunto de padrões para gerenciamento que inclui um protocolo, uma especificação de estrutura de dados, e um conjunto de objetos de dados. Este protocolo hoje já está na sua segunda versão oficial, chamada de SNMPv2. E já existem estudos para o desenvolvimento do SNMPv3. Este é o protocolo de gerência adotado como padrão para redes TCP/IP.

Pela sua complexidade, e pela lentidão do processo de padronização, este sistema de gerenciamento não é muito popular (Stallings, 1993).

O gerenciamento da rede realizado pelo protocolo SNMP, permite que uma ou mais máquinas na rede sejam designadas gerentes da rede. Esta máquina recebe informações de todas as outras máquinas da rede, chamadas agentes, e através do processamento destas informações pode gerenciar toda a rede e detectar facilmente problemas ocorridos.

As informações coletadas pela máquina gerente estão armazenadas nas próprias máquinas da rede, em uma base de dados conhecida como *Management Information Base* (MIB). Nesta base de dados estão gravadas todas as informações necessárias para o gerenciamento deste dispositivo, através de variáveis que são requeridas pela estação gerente.

Entretanto, em uma interligação de diversas redes locais, pode ser que uma rede local esteja funcionando perfeitamente, mas sem conexão com as outras redes, e, conseqüentemente, sem conexão com a máquina gerente. O ideal é implementar em alguma máquina, dentro desta rede local, um protocolo para gerenciamento que permita um trabalho *off-line*, isto é, que a rede local possa ser gerenciada, ou pelo menos tenha suas informações de gerenciamento coletadas, mesmo que estas informações não sejam enviadas instantaneamente a estação gerente.

2.5.2 RMON

Para promover uma maior e melhor expansão das tecnologias de rede, era necessário um padrão de gerenciamento de redes mais sofisticado. Com esta premissa o IETF (Internet Engineering Task Force) criou o RMON (Remote Network

Monitoring MIB) que possui como características principais a interoperabilidade independentemente de fabricante; capacidade de fornecer informações precisas a respeito das causas de falha no funcionamento normal da rede, assim como da severidade dessa falha; finalmente, o novo padrão deveria oferecer ferramentas adequadas para diagnóstico da rede. Além destas características, o padrão deveria oferecer um mecanismo pró-ativo para alertar o administrador dos eventuais problemas da rede, além de métodos automáticos capazes de coletar dados a respeito desses problemas.

Assim, o RMON tornou-se um padrão por volta de 1990. Para ampliar a capacidade do RMON foi criado o RMON II, cujo padrão está disponível nas RFCs (Request For Coments) 1757 e 1531 e apresentam padrões para redes Ethernet e Token Ring.

2.5.3 Sistemas de gerenciamento OSI

O gerenciamento no modelo *OSI* (*Open Systems Interconnect*) da *ISO* (*International Sorganization for Standartization*) baseia-se na teoria da orientação a objetos. O sistema representa os recursos gerenciados através de entidades lógicas chamadas de objetos gerenciados. Ao desenvolver uma aplicação de gerenciamento, usa-se processos distribuídos conhecidos como gerentes (que gerenciam a rede) e agentes (os que realizam as ações).

O *CMIP* (*Common Management Information Protocol*) é o protocolo de gerência do modelo OSI. Trata-se de um protocolo poderoso, robusto e abrangente (em relação ao *SNMP*), que já foi concebido com o objetivo de adequar-se à complexidade das redes . As PDUs (*Protocol Data Units*) do CMIP (que suportam as

UNIVERSIDADE TUIUTI
BIBLIOTECA
Sydnei Schaffer Emygdi Santos

primitivas do *CMIS - Common Management Information Service*) permitem a manipulação de objetos (criação, remoção), a leitura e escrita de variáveis e a notificação de eventos.

O Serviço de Informação de Gerenciamento são os serviços prestados à camada de aplicação. É orientado a conexão, necessitando de um canal virtual (uma associação) para troca de informações primitivas oferecidas para as aplicações, manipular objetos. Para a manipulação de dados gerenciais, além da *MIB, CMIS* e do *CMIP* também são definidos recursos adicionais que permitem selecionar o grupo de objetos sobre os quais se aplica uma determinada operação. O *scoping*, como é chamado este recurso, permite selecionar uma sub-árvore da hierarquia de objetos sobre os quais se realizará uma única operação. Por meio dos recursos de filtro, outra facilidade do CMIP, é possível definir um conjunto de testes aplicáveis a um grupo de instâncias de objetos (seleção por predicados), que fora anteriormente selecionado através do *scoping*. Assim sendo, é possível especificar um grupo de objetos sobre a qual se desenrolará uma operação de gerenciamento.

Na arquitetura proposta pela *ISO*, cada camada do modelo *OSI* é associada a um LME (*Layer Management Entity*) que pode interagir com a mesma para fins de gerenciamento. As LMEs são interligadas e coordenadas localmente por um *SMAE* (*System Management Application Entity*) que oferece um conjunto de primitivas para as aplicações de gerenciamento *CMIS* e se comunica com outras *SMAEs* através do *CMIP*.

As primitivas oferecidas pelo *CMIS* são utilizadas por *MIS-users*. Um *MIS-user* pode desempenhar o papel de gerente, agente ou ambos, permitindo a construção de uma hierarquia de gerentes, distribuindo a responsabilidade da

gerência por vários pontos, possibilitando, inclusive a redução do tráfego de mensagens entre agentes/gerente.

2.6 NECESSIDADES DO GERENCIAMENTO DE REDES

Por menor e mais simples que seja uma rede de computadores, precisa ser gerenciada, a fim de garantir, aos seus usuários, a disponibilidade de serviços a um nível de desempenho aceitável.

À medida que a rede cresce, aumenta a complexidade de seu gerenciamento, forçando a adoção de ferramentas automatizadas para a sua monitoração e controle.

A adoção de um software de gerenciamento não resolve todos os problemas da pessoa responsável pela administração da rede. Geralmente o usuário de um software de gerenciamento espera muito dele e, conseqüentemente, fica frustrado quanto aos resultados que obtém. Por outro lado, esses mesmos softwares quase sempre são sub-utilizados, isto é, possuem inúmeras características inexploradas ou utilizadas de modo pouco eficiente. Para gerenciar um recurso, é necessário conhecê-lo muito bem e visualizar claramente o que este recurso representa no contexto da rede.

O investimento em um software de gerenciamento pode ser justificado pelos seguintes fatores:

- As redes e recursos de computação distribuídos estão se tornando vitais para a maioria das organizações. Sem um controle efetivo, os recursos não proporcionam o retorno que a corporação requer.

- O contínuo crescimento da rede em termos de componentes, usuários, interfaces, protocolos e fornecedores ameaçam o gerenciamento com perda de controle sobre o que está conectado na rede e como os recursos estão sendo utilizados.
- Os usuários esperam uma melhoria dos serviços oferecidos (ou no mínimo, a mesma qualidade), quando novos recursos são adicionados ou quando são distribuídos.
- Os recursos computacionais e as informações da organização geram vários grupos de aplicações de usuários com diferentes necessidades de suporte nas áreas de desempenho, disponibilidade e segurança. O gerente da rede deve atribuir e controlar recursos para balancear estas várias necessidades.
- À medida que um recurso fica mais importante para a organização, maior fica a sua necessidade de disponibilidade. O sistema de gerenciamento deve garantir esta disponibilidade.
- A utilização dos recursos deve ser monitorada e controlada para garantir que as necessidades dos usuários sejam satisfeitas a um custo razoável.

Além desta visão qualitativa, uma separação funcional de necessidades no processo de gerenciamento foi apresentada pela ISO, como parte de sua especificação de Gerenciamento de Sistemas OSI. Esta divisão funcional foi adotada pela maioria dos fornecedores de sistemas de gerenciamento de redes para descrever as necessidades de gerenciamento: Falhas, Desempenho, Configuração, Contabilização e Segurança.

- Gerenciamento de Falhas: Detecção, correção e prevenção de falhas;
- Gerenciamento de Desempenho: Redimensionamento da rede e de seus componentes, coletar informações estatísticas;

- Gerenciamento de Contabilização: Verificação da alocação de recursos e determinar os custos envolvidos, determinar limites de uso e custos;
- Gerenciamento de Configuração: Visualização e alteração da configuração da rede e seus componentes, Gerenciar o ciclo de vida do sistema e sua configuração associada;
- Gerenciamento de Segurança: Proteção contra ataques aos componentes da rede, gerenciar os mecanismos e procedimentos que proporcionam a proteção aos recursos da rede, garantir a manutenção da política de segurança.

2.7 HISTÓRICO DO GERENCIAMENTO DE REDES

Quando em 1986 reuniu-se, pela primeira vez, o Grupo de Trabalho sobre gerenciamento de Redes do Comitê Técnico em Comunicação de dados IFIP (*International Federation for Information Procesing*) havia apenas o consenso sobre a necessidade de gerenciamento. Cerca de 20 pessoas reunidas em Dallas, provenientes de diversos países, sequer concordavam sobre o escopo do gerenciamento de rede. Para alguns dos representantes, incorporasse apenas as três camadas inferiores da arquitetura OSI (pois era com o que estavam acostumados a trabalhar), para os outros o gerenciamento de redes devia englobar as sete camadas. Percebia-se claramente que cada fornecedor tinha construído uma arquitetura proprietária de gerenciamento para seus produtos e tinha dificuldade de impingi-la aos clientes, ao lado de outros fornecedores. Já se falava na oportunidade sobre o gerenciamento OSI, embora muitos tenham encarado com certo ar de dúvida aquela alternativa.

A abordagem clássica para integrar o gerenciamento de redes era baseada em arquitetura proprietárias. Para que pudessem funcionar como elemento de integração, os arquitetos de tais soluções incorporaram nelas uma abertura para agregar a informação de gerenciamento de sistema de outros fornecedores. A IBM®, por exemplo, com o conceito de *focal point* abriu esta porta para integrar outros sistemas de gerenciamento ao *Netview*, principalmente por interesse próprio, uma vez que a aquisição da RDLM (fabricante de PABX) levou a esta necessidade. Módulos para traduzir o fluxo de informação de gerenciamento de um esquema para outros tinham de ser constituídos e podiam ser implantados em vários pontos. Poderia ser um servidor de rede Novell®, com um módulo interno capaz de gerar os "vetores de alerta" esperados pelo Netview. Como segunda opção poderia ser a solução para integrar o gerenciamento de um PABX digital em que a tradução seria feita em um PC que receberia as mensagens de gerenciamento de um lado e as traduziria, quando possível, para o outro. A terceira abordagem seria para o caso em que um roteador fosse o diagnóstico gerenciado e que utilizasse um protocolo padrão de fato na indústria, tal como o SNMP, com a conversão feita internamente no computador gerenciador.

Dentro dos problemas decorrentes desta solução, pode-se destacar principalmente a limitação imposta pelo fato de somente usar opções gerenciamento (dados recebidos e comandos veiculáveis) que tinham similar na arquitetura proprietária do fornecedor do computador gerenciador. Opções de interação propiciadas pelos dispositivos gerenciados podiam não ser aproveitadas simplesmente pela falta de condições de mapeá-las para uma forma passível de reconhecimento pelo computador gerenciador. Em decorrência, os dispositivos gerenciados providos pelo mesmo fornecedor do computador gerenciador

"pareciam" mais facilmente gerenciáveis. Para não parecerem diminuídos sob este prisma, muitos fornecedores não se mostravam entusiasmados em cooperar para tornar seus produtos gerenciáveis por um computador gerenciador de outro fabricante.

Esta abordagem foi adotada por alguns fornecedores no mercado, como a IBM® e a DEC®, mas cada vez mais crescia o desejo por um sistema de gerenciamento independente de fornecedor que pudesse rodar em uma máquina dedicada, de modo a não sobrecarregar nem prejudicar o atendimento dos serviços normais a serem executados no *mainframe*. A AT&T® também entrou no cenário, definindo uma arquitetura de gerenciamento e se propondo a gerenciar as redes de seus clientes de telecomunicações.

Criando o impasse, uma solução alternativa teria de ser buscada, implicando a agregação de esforços que levassem a uma solução mais universal e padronizada. Obviamente, tal solução deveria englobar os serviços de gerenciamento mais importantes e relevantes, além de formalizar a interação entre os dispositivos gerenciados e os gerenciadores. A ISO tomou a bandeira e o esquema básico da arquitetura de gerenciamento de rede foi adicionado ao modelo de referência ISO/OSI em 1989.

A colaboração entre a ISO/IEC (*International Organization for Standardization / International Eletrotechnical Committee*) resultou na série de documentos X.700, cujo objetivo maior é criar condições para o desenvolvimento de produtos de gerenciamento de redes de computadores e sistema de comunicações heterogêneos.

Porém, o embate das forças dominantes no cenário internacional dificultou a estabilização dos detalhes para a operacionalização do modelo de gerenciamento.

Anos se passaram sem que os documentos atingissem o estágio do padrão ISO internacional. As implantações, baseadas em interpretações da documentação disponível, começaram a aparecer e, em 1989, percebendo a necessidade de acordos que assegurassem a interoperabilidade das implementações, os fornecedores começaram a reunir-se em associações como a ISO/NM Fórum, para buscar um acordo que viabilizasse a definição de um conjunto de opções de implantação capaz de assegurar a interoperabilidade dos sistemas de gerenciamento. Outro grupo foi criado sob a tutela do NIST (*National Institute of Standards and Technology*) dos Estados Unidos para atender às necessidades do governo americano, que já havia determinado, através de seu documento GOSIP (*Government OSI Profile*), que as soluções de redes a serem adquiridas deveriam atender às recomendações ISO/IEC. Este trabalho resultou no GNMP (*Government Network Management Profile*), cuja versão 1, de 30 de julho de 1992, constitui a referência que todas as agências do governo federal dos Estados Unidos devem usar ao adquirir funções e serviços de gerenciamento de rede.

O primeiro dos protocolos de gerência de rede foi o SGMP (*Simple Gateway Monitoring Protocol*) que surgiu em novembro 1987. Entretanto, o SGMP era restrito à monitoração de *gateways*. A necessidade crescente de uma ferramenta de gerenciamento de rede mais genérica fez emergirem mais algumas abordagens:

High-Level Entity Management System – HEMS – generalização do HMP – *Host Management Protocol*;

SNMP – *Simple Network Management Protocol* – um melhoramento do SGMP;

CMOT – (CMIP *over* TCP/IP) uma tentativa de incorporar o máximo possível o protocolo (CMIP), serviços e estrutura de base de dados que estava sendo padronizada pela ISO para gerenciamento de redes.

No início de 1988 a IAB (*Internet Architecture Board*) revisou os protocolos e escolheu o SNMP como uma solução de curto prazo e o CMOT como solução de longo prazo para o gerenciamento de redes. O sentimento era que, em um período de tempo razoável, as instalações migrariam do *TCP* para protocolos baseados em OSI. Entretanto, como a padronização do gerenciamento baseado no modelo OSI apresentava muita complexidade de implementação e o SNMP, devido à sua simplicidade, foi amplamente implementado nos produtos comerciais, o SNMP tornou-se um padrão de fato. Posteriormente, pela existência de lacunas funcionais (devido exatamente à simplicidade do SNMP), foram definidas novas versões do protocolo SNMP chamadas de SNMPv2 e SNMPv3, e o SNMP original ficou conhecido como SNMPv1.

A primeira versão da arquitetura de gerenciamento SNMP foi definida no RFC 1157 de maio de 1990.

O RFC 1157 define ainda três objetivos a serem alcançados pelo SNMP: minimizar o número e complexidade das funções de gerenciamento, ser flexível o suficiente para permitir expansões futuras e ser independente da arquitetura e mecanismo dos dispositivos gerenciados.

A definição das informações de gerenciamento requer não apenas profundo conhecimento da área específica em foco, mas também do modelo de gerenciamento.

2.8 SOFTAWARES DE GERENCIAMENTO DE REDES

2.8.1 HP *Open View*

Plataforma SNMP, contém programas executáveis que suportam as chamadas do Kit do Desenvolvedor SNMP. O HP *Open View Windows*, descobre o endereço IP automaticamente e monta o *layout* (isto é, descobre um nodo de rede *automaticamente e cria um mapa incluindo-o). Utiliza também, biblioteca de execução SNMP, manuseio de traps e uma MIB* acessível ao usuário final.

A Plataforma SNMP descobre e monitora dinamicamente padrões como: endereçamento IP, SNMP MIB-II e dispositivos SNMP. Os produtos de gerenciamento SNMP proporcionam um ambiente de programação estável e interoperabilidade com outros sistemas baseados em padrões.

2.8.2 Net*View*

É uma ferramenta de gerenciamento de sistema e de rede que proporciona funções de configuração, falha e desempenho. Possui uma plataforma de gerenciamento de rede aberto que habilita a integração de aplicações SNMP (*Simple Network Management Protocol*) e CMIP (*Common Management Information Protocol*).

A plataforma *NetView* foi desenvolvida para vários ambientes operacionais : AIX, VMS, Sun Solaris, AS/400. Este trabalho foi baseado na utilização do *NetView* no sistema operacional AIX.

2.8.3 *Spectrum*

É uma plataforma de gerenciamento que fornece independência de protocolo e gerenciamento de dispositivos diferentes fabricantes para a integração completa de sistemas para grandes empresas. Administra redes ativamente, localiza e corrige problemas, e informa aos gerentes da rede sobre mudanças de *status*. Diferente de outras plataformas, *SPECTRUM* realmente não tem problemas para administrar uma rede. Através do uso da Tecnologia de Modelação Indutiva (IMT), uma forma de inteligência artificial, *SPECTRUM* é capaz de resolver problemas sobre suas próprias soluções iniciadas. A IMT permite ao *SPECTRUM* criar um modelo de cada entidade na rede, incluindo cabos físicos, aparelhos de redes, topologias e aplicações. Cada modelo é inteligente, até mesmo se a entidade que está representada não é inteligente.

2.8.4 *SunNet Manager*

É uma coleção de ferramentas para gerenciamento de redes. Estas ferramentas provêm uma plataforma comum como base para todas as funções de gerência numa arquitetura uniforme, além de disponibilizar uma interface de usuário.

A arquitetura *SunNet Manager* ainda provê um sistema de gerenciamento de redes que é independente dos protocolos inferiores. Os serviços oferecidos não fazem quaisquer suposições sobre os protocolos específicos de rede (TCP/IP ou UDP/IP) usados para suprir as funções de transporte de dados.

A versão 1.0 do *SunNet Manager* é baseada no ambiente ONC (*Open Network Computing*) da *Sun Microsystems*, mas já existe a versão 2.2.2 conhecida por *Solstice SunNet Manager*.

3 METODOLOGIA

3.1 LINGUAGEM DE PROGRAMAÇAO

Como base para o desenvolvimento deste protótipo foi utilizada a linguagem de programação Delphi na versão 7. Esta linguagem foi escolhida por ser de alto nível e possuir os recursos RAD (*Rapid Application Development*) – Desenvolvimento Rápido de Aplicativo, utilizando a linguagem Pascal como padrão, e tendo diversas comunidades de auxilio e principalmente por utilizar conexões com diversos banco de dados.

Uma linguagem que gera aplicativos executáveis rápidos e com tamanhos reduzidos, comparados com outras linguagens.

Neste protótipo também foi utilizado o banco de dados *Interbase*, devido sua interoperação com o Delphi. A distribuição deste banco de dados é gratuita. È utilizado o *FireBird*, um SGBD (Sistema Gerenciador de Banco de Dados) que é um produto de Código Livre desenvolvido sobre licensa *Interbase License*.

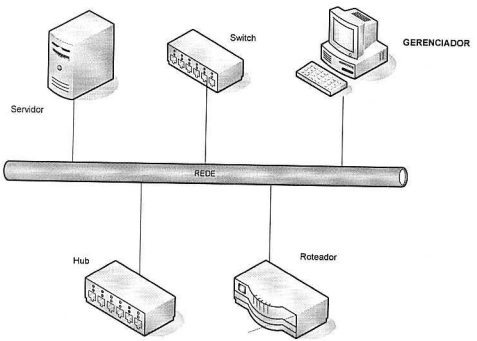
O *FireBird* oferece uma excelente solução como sistema gerenciador de bancos de dados de alta performance. Além disso o *FireBird* lhe oferece excelente concorrência de acesso, alta performance, e uma poderosa linguagem com suporte a *stored procedures* e *triggers* de bancos.

3.2 PRÉ-REQUISITOS

Para se utilizar o protótipo é necessário a instalação das bibliotecas PING (Anexo 1) e ICMP (Anexo 2), desenvolvidas por François Piette e disponibilizada gratuitamente sem qualquer restrição de uso. Estas duas bibliotecas foram a base para o desenvolvimento deste protótipo.

3.3 ESTRUTURA DE FUNCIONAMENTO DO PROTÓTIPO

FIGURA 6: Estrutura Geral do Protótipo



O protótipo, através de uma rede TCP/IP, executa testes dos acessos até os ativos de rede, através do envio de um pedido de eco ICMP e da espera por uma resposta.

3.4 PRINCÍPIO DE FUNCIONAMENTO DO PROTÓTIPO

O protótipo está baseado no comando *ping* do sistema operacional. O mesmo envia para os ativos de rede de destino um pedido de resposta para suas requisições, as quais são mostradas na tela e armazenadas em banco de dados para futuras análises. O protótipo continuará a verificação até a interrupção pelo usuário. Obtendo ou não as respostas solicitadas o protótipo armazenará os resultados no banco de dados.

Este protótipo permite realizar a verificação específica de um ativo de rede ou de um grupo. Os resultados dos testes podem ser visualizados e impressos selecionando um ativo de rede ou todos ativos de rede por período.

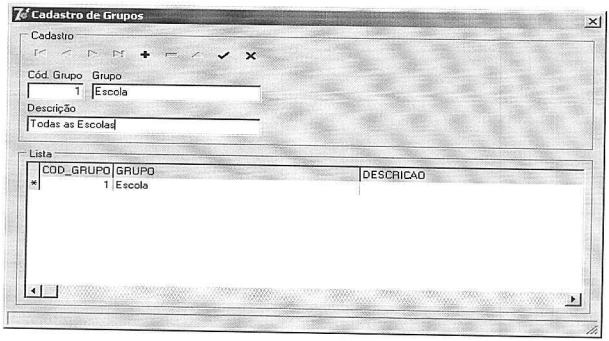
3.5 MANUAL DO USUÁRIO

O Programa pingtst.exe é um protótipo desenvolvido em Delphi, e utiliza o banco de dados *Interbase*. É necessário que o computador esteja ligado em algum ativo de rede (modem, *Hub*, *Switch*, Roteador ou outra estação de trabalho).

Configurações Mínimas:
Sistema Operacional: Windows (2000 e XP)
Estação: Pentium 233 ou superior
Memória RAM: 128MB
Rede: TCP/IP

Cadastro de Grupos

Menu Cadastro / Grupos



Clique no ✚ para adicionar um registro em branco, depois preencha os campos:

- Código
- Nome do Grupo
- Descrição

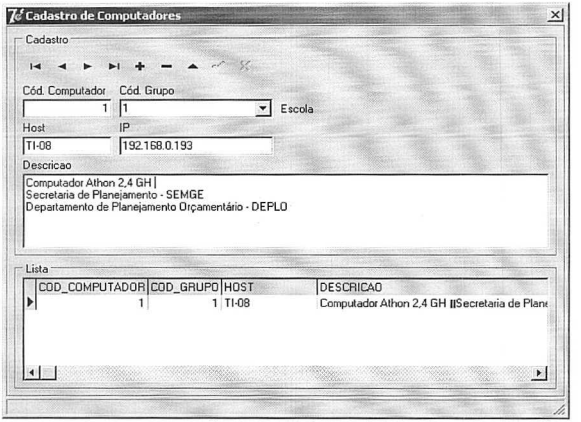
O campos Código e Nome do Grupo são obrigatórios

Para confirmar clique no ✓

Para Cancelar clique ✖

Cadastro Ativo de Rede

Menu Cadastro / Ativo de Rede



Clique no ✚ para adicionar um registro em branco, depois preencha os campos:

- Código
- Grupo que este Ativo de rede fará parte
- Nome do Ativo de Rede (*Host*) e Descrição

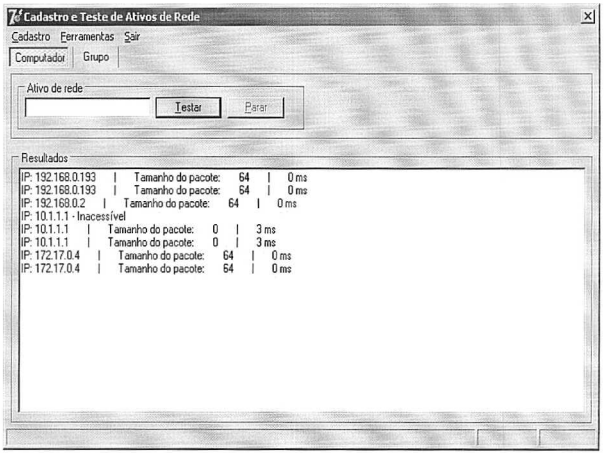
Todos campos acima são obrigatórios

Para confirmar clique no ✓

Para Cancelar clique ✖

Testando um Ativo de Rede

Verificar se está ativo o item Computador


Cadastro e Teste de Ativos de Rede
Cadastro Ferramentas Sair
Computador Grupo
Ativo de rede
Testar
Parar
Resultados
IP: 192.168.0.193 | Tamanho do pacote: 64 | 0 ms
IP: 192.168.0.193 | Tamanho do pacote: 64 | 0 ms
IP: 192.168.0.2 | Tamanho do pacote: 64 | 0 ms
IP: 10.1.1.1 - Inacessível
IP: 10.1.1.1 | Tamanho do pacote: 0 | 3 ms
IP: 10.1.1.1 | Tamanho do pacote: 0 | 3 ms
IP: 172.17.0.4 | Tamanho do pacote: 64 | 0 ms
IP: 172.17.0.4 | Tamanho do pacote: 64 | 0 ms

Preencher a lacuna em branco Ativo de Rede com:

- Endereço de rede do ativo de rede ou
- Nome do ativo de rede.

Testando a disponibilidade do ativo de rede.

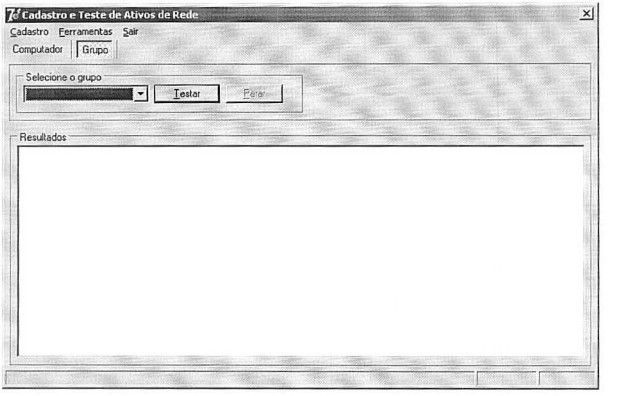
Clicar no botão Testar.

O programa ficará testando se o ativo de rede está disponível ou não e registrando em banco de dados os resultados.

Para Parar deverá obrigatoriamente clicar no botão Parar.

Testando um Grupo

Verificar a disponibilidade dos ativos de rede cadastrados no grupo escolhido



Escolher o grupo a ser testado

Testando a disponibilidade do ativo de rede no grupo.

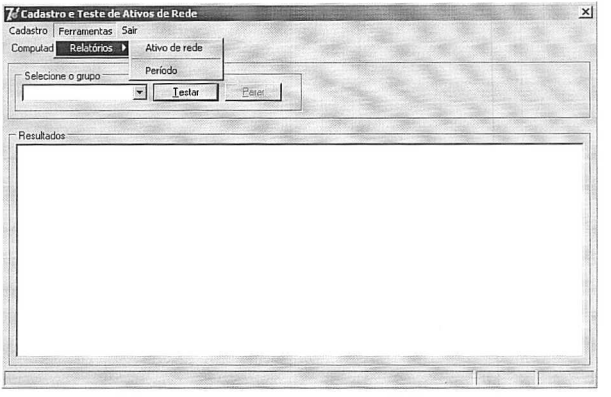
Clicar no botão Testar.

O programa ficará testando se os ativos de rede estão disponíveis ou não e registrando em banco de dados os resultados.

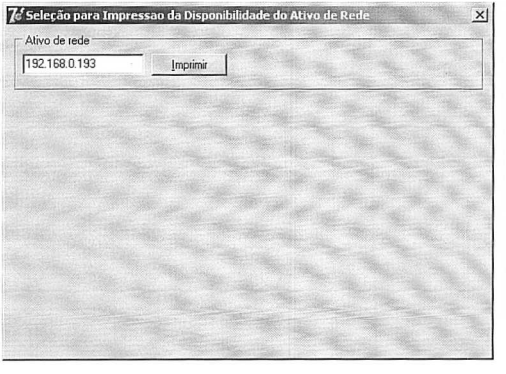
Para Parar deverá obrigatoriamente clicar no botão Parar.

Listando os resultados por Grupo

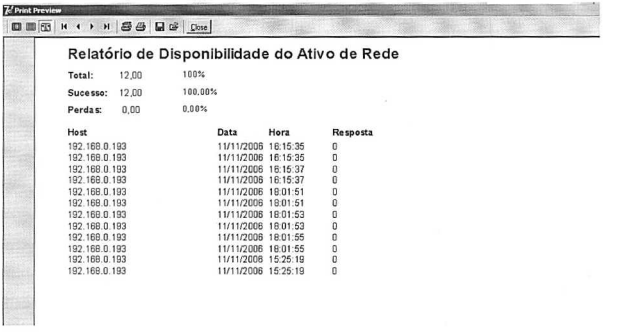
Poderá selecionar as opções conforme figura a seguir



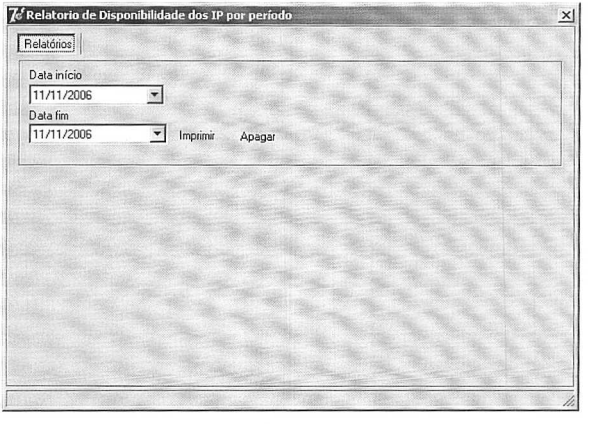
Opção Ativo de rede



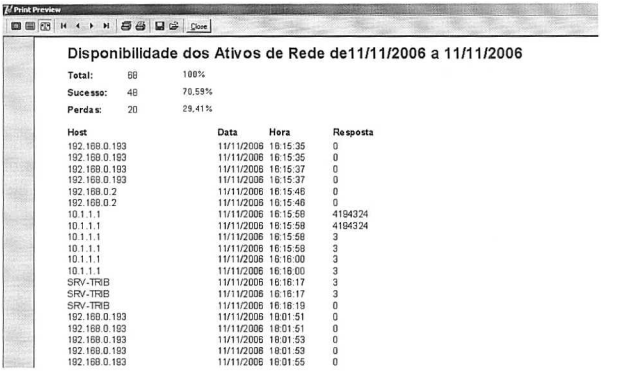
Após colocar o endereço do ativo de rede e clicar no botão Imprimir obterá um relatório em tela que poderá ser impresso.



Opção Período



Após selecionar o período que deseja-se verificar clicar no botão Imprimir e terá um relatório das disponibilidades de todos os ativos de rede testados no período.


Print Preview

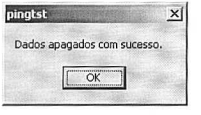
Close

Disponibilidade dos Ativos de Rede de11/11/2006 a 11/11/2006

Total:	68	100%
Sucesso:	48	70,59%
Perdas:	20	29,41%

Host	Data	Hora	Resposta
192.168.0.193	11/11/2006	16:15:35	0
192.168.0.193	11/11/2006	16:15:35	0
192.168.0.193	11/11/2006	16:15:37	0
192.168.0.193	11/11/2006	16:15:37	0
192.168.0.2	11/11/2006	16:15:46	0
192.168.0.2	11/11/2006	16:15:46	0
10.1.1.1	11/11/2006	16:15:58	4194324
10.1.1.1	11/11/2006	16:15:58	4194324
10.1.1.1	11/11/2006	16:15:58	3
10.1.1.1	11/11/2006	16:15:58	3
10.1.1.1	11/11/2006	16:16:00	3
10.1.1.1	11/11/2006	16:16:00	3
SRV-TRIB	11/11/2006	16:16:17	3
SRV-TRIB	11/11/2006	16:16:17	3
SRV-TRIB	11/11/2006	16:16:19	0
192.168.0.193	11/11/2006	18:01:51	0
192.168.0.193	11/11/2006	18:01:51	0
192.168.0.193	11/11/2006	18:01:53	0
192.168.0.193	11/11/2006	18:01:53	0
192.168.0.193	11/11/2006	18:01:55	0

Após selecionar o período que e desejar eliminar estas informações poderá clicar no botão Apagar e será eliminado estas informações do banco de dados com a mensagem a seguir no final da operação.

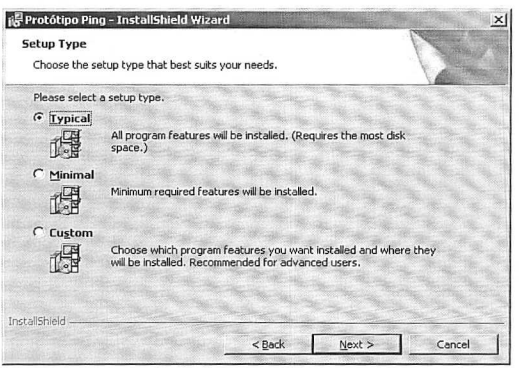


3.6 MANUAL DE INSTALAÇÃO

Executar o arquivo pingtst.exe e pressionar em *Next* (Próximo)

Protótipo Ping - InstallShield Wizard
Welcome to the InstallShield Wizard for Protótipo Ping
The InstallShield(R) Wizard will install Protótipo Ping on your computer. To continue, click Next.
WARNING: This program is protected by copyright law and international treaties.
< Back
Next >
Cancel

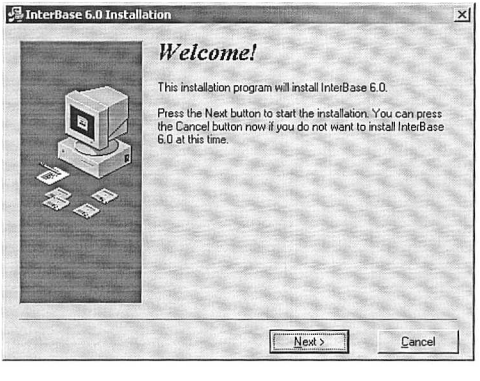
Selecionar a opção *Typical* (Típica) e pressionar em *Next*


Protótipo Ping - InstallShield Wizard
Setup Type
Choose the setup type that best suits your needs.
Please select a setup type.
Typical
All program features will be installed. (Requires the most disk space.)
Minimal
Minimum required features will be installed.
Custom
Choose which program features you want installed and where they will be installed. Recommended for advanced users.
InstallShield
< Back
Next >
Cancel

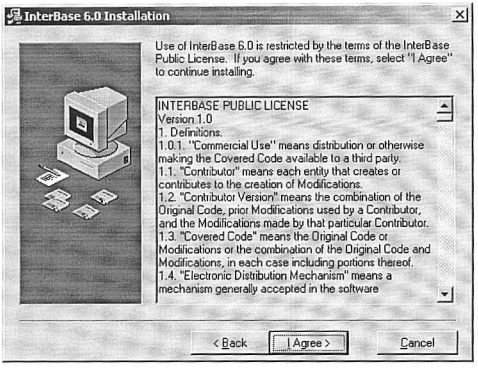
Pressionar em *Finish* (Finalizar) para término da instalação

Protótipo Ping - InstallShield Wizard
InstallShield Wizard Completed
Setup has finished installing Protótipo Ping.
Some program files might have been updated since you purchased your copy of Protótipo Ping.
Your Internet connection can be used to make sure that you have the latest updates.
Yes, check for program updates (Recommended) after the setup completes.
< Back
Finish
Cancel

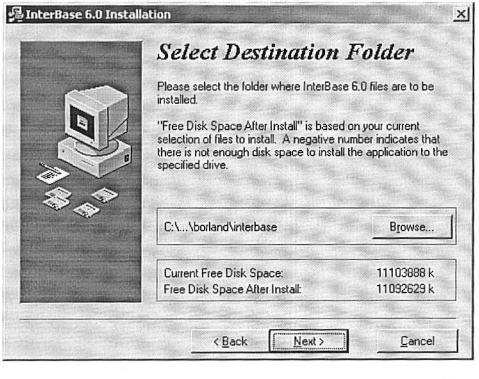
Executar o aplicativo instalador do *Interbase* 6.0 ib_wi_os_020204.exe e pressionar em *Next*


InterBase 6.0 Installation
Welcome!
This installation program will install InterBase 6.0.
Press the Next button to start the installation. You can press the Cancel button now if you do not want to install InterBase 6.0 at this time.
Next >
Cancel

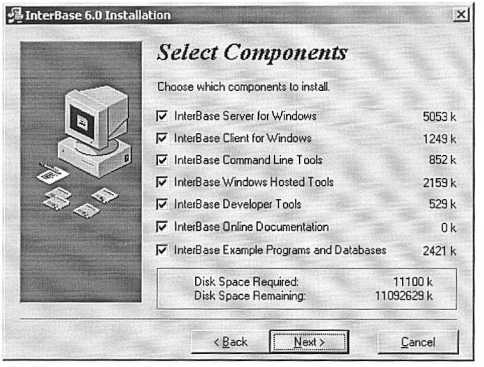
Pressionar em *I Agree* (Eu Concordo)


InterBase 6.0 Installation
Use of InterBase 6.0 is restricted by the terms of the InterBase Public License. If you agree with these terms, select "I Agree" to continue installing.
INTERBASE PUBLIC LICENSE
Version 1.0
1. Definitions.
1.0.1. "Commercial Use" means distribution or otherwise making the Covered Code available to a third party.
1.1. "Contributor" means each entity that creates or contributes to the creation of Modifications.
1.2. "Contributor Version" means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.
1.3. "Covered Code" means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.
1.4. "Electronic Distribution Mechanism" means a mechanism generally accepted in the software
< Back
I Agree >
Cancel

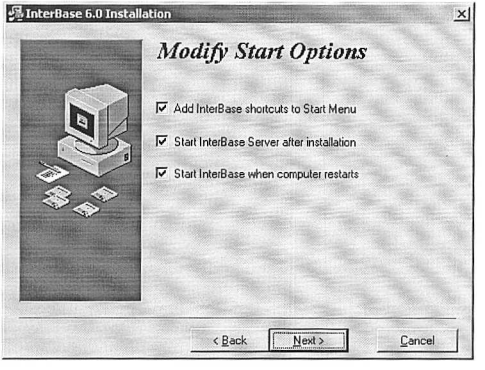
Pressionar em *Next* para continuar a instalação


InterBase 6.0 Installation
Select Destination Folder
Please select the folder where InterBase 6.0 files are to be installed.
"Free Disk Space After Install" is based on your current selection of files to install. A negative number indicates that there is not enough disk space to install the application to the specified drive.
C:\...\borland\interbase
Browse...
Current Free Disk Space:
11103888 k
Free Disk Space After Install:
11092629 k
< Back
Next >
Cancel

Pressionar em *Next* para continuar a instalação

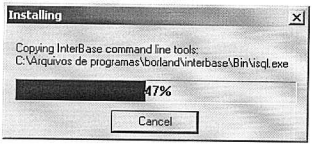

InterBase 6.0 Installation
Select Components
Choose which components to install.
InterBase Server for Windows 5053 k
InterBase Client for Windows 1249 k
InterBase Command Line Tools 852 k
InterBase Windows Hosted Tools 2159 k
InterBase Developer Tools 529 k
InterBase Online Documentation 0 k
InterBase Example Programs and Databases 2421 k
Disk Space Required: 11100 k
Disk Space Remaining: 11092629 k
< Back
Next >
Cancel

Pressionar em *Next* para continuar a instalação

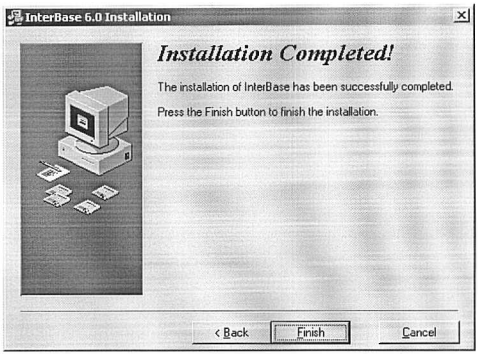


Pressionar em *Next* para continuar a instalação

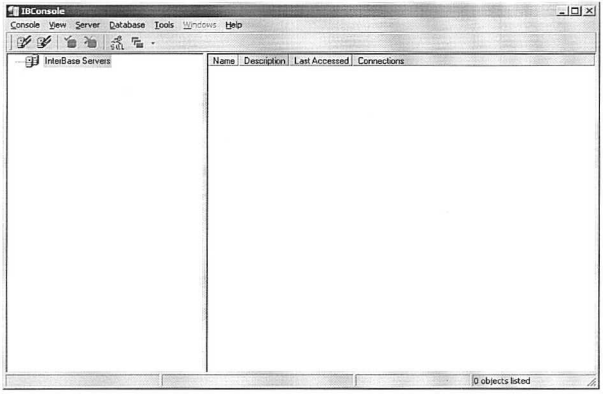
InterBase 6.0 Installation
Ready to Install!
You are now ready to install InterBase 6.0.
Press the Next button to begin the installation or the Back button to reenter the installation information.
< Back
Next >
Cancel



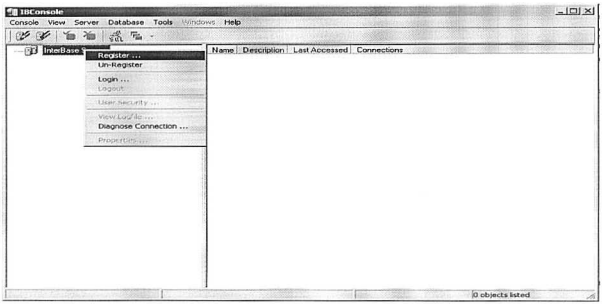
Pressionar em *Finish* (Finalizar) para término da instalação



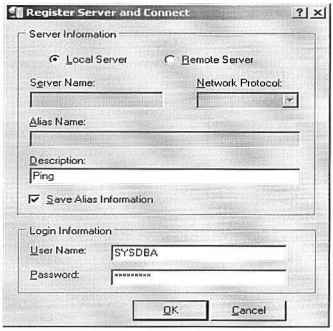
Selecione no botão do Menu Iniciar / Programas / interbase 6.0 / ibconsole



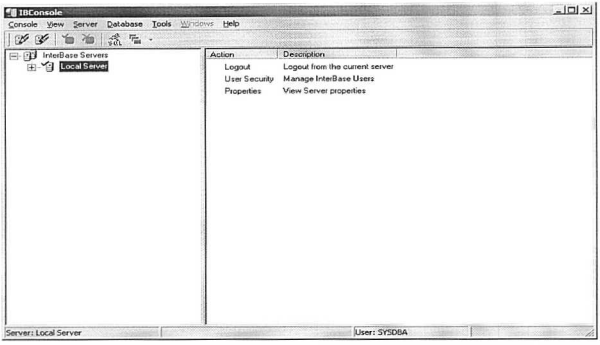
Selecione InterBase Servers / Register



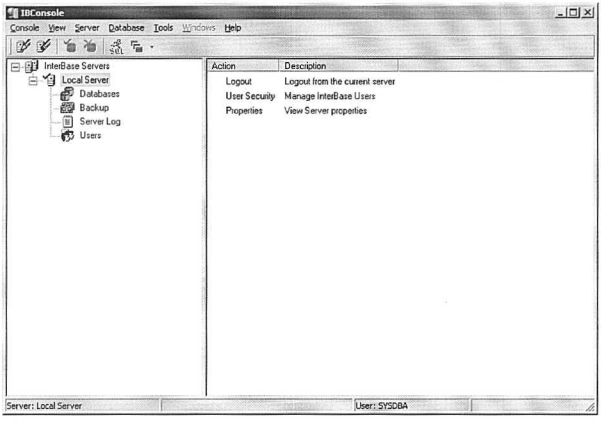
Selecione a opção *Local Server* (Servidor Local), no Campo *Description* (Descrição) escreva Ping e selecione a opção *Save Alias Information*. Preencher no campo *User Name* (Nome do Usuário) o valor SYSDBA e no campo *Password* (senha) o valor *masterkey* (minúsculo) e pressione OK para continuar a instalação.



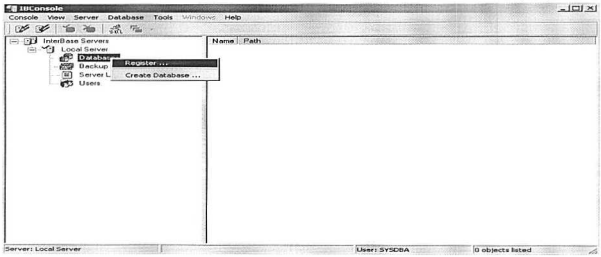
Pressionar duas vezes *Local Server*



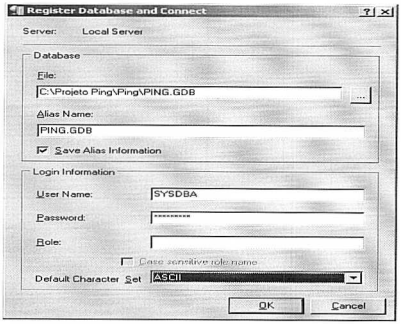
Serão mostradas opções após pressionar *Local Server*



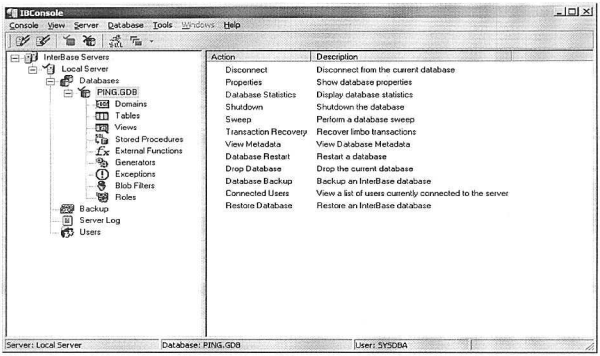
Selecione *Databases / Register*



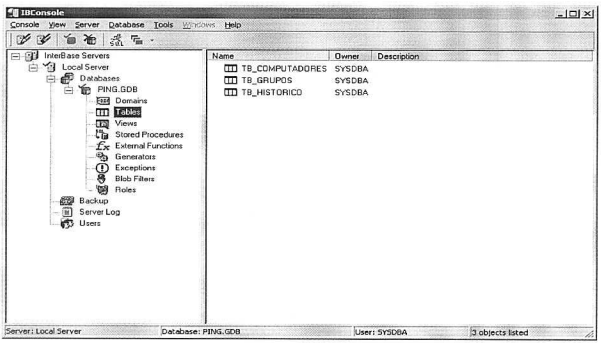
No campo *File* (arquivo) preencha com o valor C:\Projeto Ping\PING\PING.GDB, no campo *Alias Name* preencha com o valor PING.GDBe selecione a opção *Save Alias Information*. Preencher no campo *User Name* o valor SYSDBA, no campo *Password* o valor *masterkey* (minúsculo) e no campo *Default Character* Set o valor ASCII e pressione OK para continuar a instalação



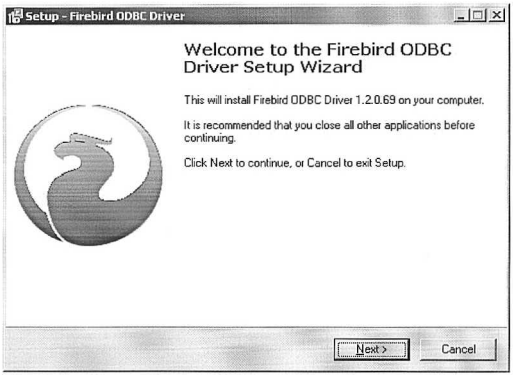
Pressionar duas vezes *DataBases* após pressionar duas vezes PING.GDB



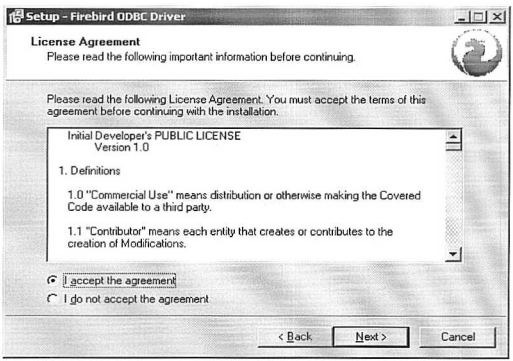
Pressionar duas vezes *Tables*, será mostrado as tabelas utilizadas. O banco de dados já está instalado



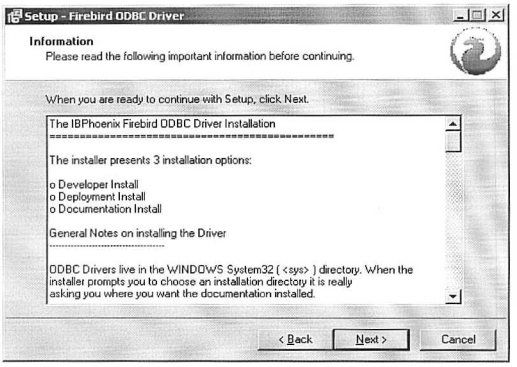
Pressionar duas vezes o arquivo Firebird_ODBC_1.2.0.69-Win32.exe e pressionar em *Next* para continuar a instalação.


Setup - Firebird ODBC Driver
Welcome to the Firebird ODBC Driver Setup Wizard
This will install Firebird ODBC Driver 1.2.0.69 on your computer.
It is recommended that you close all other applications before continuing.
Click Next to continue, or Cancel to exit Setup.
Next >
Cancel

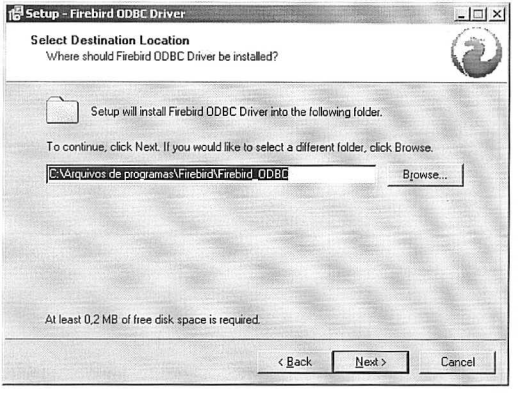
Marcar a opção *I accept the agreement* e pressionar *Next* para continuar a instalação


Setup - Firebird ODBC Driver
License Agreement
Please read the following important information before continuing.
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.
Initial Developer's PUBLIC LICENSE
Version 1.0
1. Definitions
1.0 "Commercial Use" means distribution or otherwise making the Covered Code available to a third party.
1.1 "Contributor" means each entity that creates or contributes to the creation of Modifications.
I accept the agreement
I do not accept the agreement
< Back
Next >
Cancel

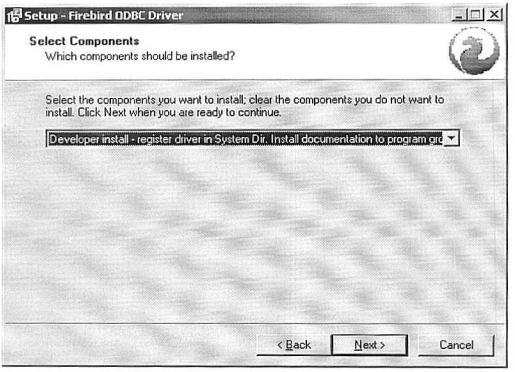
Pressionar em *Next* para continuar a instalação.


Setup - Firebird ODBC Driver
Information
Please read the following important information before continuing.
When you are ready to continue with Setup, click Next.
The IBPhoenix Firebird ODBC Driver Installation
The installer presents 3 installation options:
o Developer Install
o Deployment Install
o Documentation Install
General Notes on installing the Driver
ODBC Drivers live in the WINDOWS System32 (<sys>) directory. When the installer prompts you to choose an installation directory it is really asking you where you want the documentation installed.
< Back
Next >
Cancel

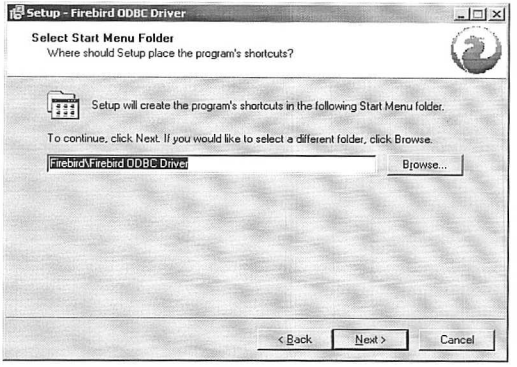
Pressionar em *Next* para continuar a instalação.


Setup - Firebird ODBC Driver
Select Destination Location
Where should Firebird ODBC Driver be installed?
Setup will install Firebird ODBC Driver into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Arquivos de programas\Firebird\Firebird_ODBC
Browse...
At least 0,2 MB of free disk space is required.
< Back
Next >
Cancel

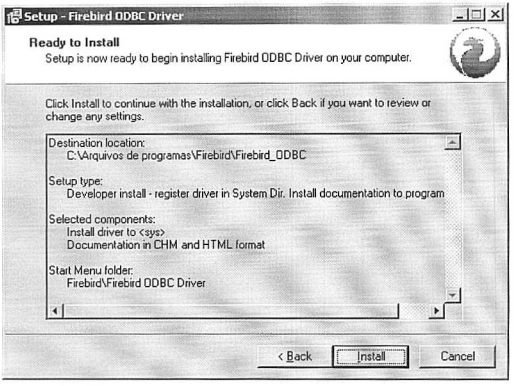
Pressionar em *Next* para continuar a instalação.


Setup - Firebird ODBC Driver
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Developer install - register driver in System Dir, Install documentation to program grc
< Back
Next >
Cancel

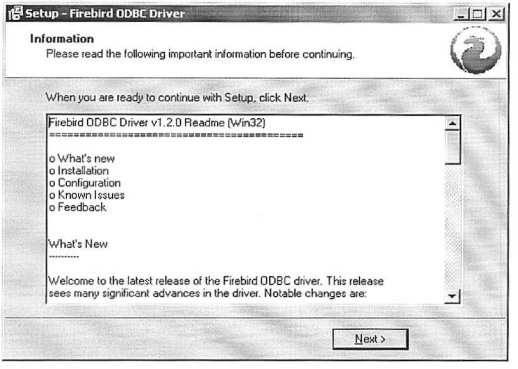
Pressionar em *Next* para continuar a instalação.


Setup - Firebird ODBC Driver
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Firebird\Firebird ODBC Driver
Browse...
< Back
Next >
Cancel

Pressionar em *Install* para continuar a instalação.


Setup - Firebird ODBC Driver
Ready to Install
Setup is now ready to begin installing Firebird ODBC Driver on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Destination location:
C:\Arquivos de programas\Firebird\Firebird_ODBC
Setup type:
Developer install - register driver in System Dir. Install documentation to program
Selected components:
Install driver to <sys>
Documentation in CHM and HTML format
Start Menu folder:
Firebird\Firebird ODBC Driver
< Back
Install
Cancel

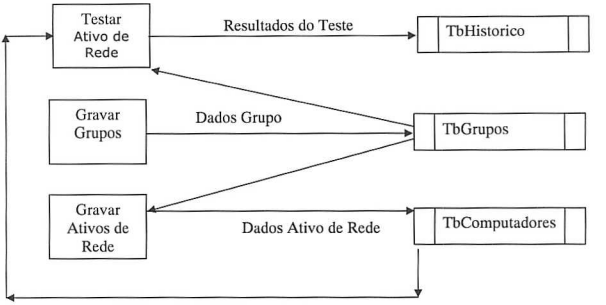
Pressionar em *Next* para continuar a instalação.


Setup - Firebird ODBC Driver
Information
Please read the following important information before continuing.
When you are ready to continue with Setup, click Next.
Firebird ODBC Driver v1.2.0 Readme (Win32)
o What's new
o Installation
o Configuration
o Known Issues
o Feedback
What's New
Welcome to the latest release of the Firebird ODBC driver. This release sees many significant advances in the driver. Notable changes are:
Next >

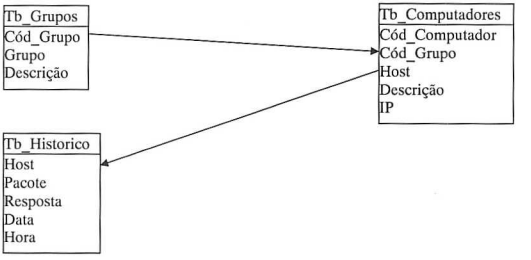
Pressionar em *Finish* para término da instalação.

Setup - Firebird ODBC Driver
Completing the Firebird ODBC Driver Setup Wizard
Setup has finished installing Firebird ODBC Driver on your computer. The application may be launched by selecting the installed icons.
Click Finish to exit Setup.
View Readme.txt
< Back
Finish

3.7 DFD – DIAGRAMA DE FLUXO DE DADOS



3.8 DER – DIAGRAMA DE ENTIDADE E RELACIONAMENTO



4 TESTES

Após o desenvolvimento do protótipo, realizou-se testes nas opções de ativo de rede e grupo, bem como a visualização dos relatóriosgerados.

Na opção ativo de rede o teste é executado utilizando um endereço IP ou seu "nome".

Na opção grupo, realizou-se testes com 30 ativos de redes cadastrados. Para os testes cadastrou-se ativos de redes utilizando endereços IP e "nomes".

Após a realização dos testes, verificou-se o correto funcionamento na geração dos relatórios.

Em todos os testes obteve-se o comportamento e os resultados esperados do protótipo.

5 CONCLUSÃO

Durante o desenvolvimento do projeto fez se necessário a ampliação sobre o conhecimento na área de redes e de programação através de pesquisas, bem como foram possíveis a utilização dos conhecimentos adquiridos em sala de aula.

Utilizou-se para o desenvolvimento deste protótipo a linguagem de programação Delphi. Foi observado ser uma linguagem robusta, a qual possibilita uma boa definição gráfica, além de ser de fácil aprendizado e utilização. Foi verificado que esta linguagem gera aplicativos de tamanhos reduzidos, logo de rápida execução. Com relação ao banco de dados Interbase, foi utilizado devido a sua interoperação com o Delphi e por ser de distribuição gratuita. Porém foram observadas diferenças na execução dos comandos SQL entre Paradox e Interbase. A correção destas diferenças entre comandos foi realizada através de pesquisas e fóruns na internet.

O protótipo como um todo obteve uma boa performance. A sua operação foi de forma continua, ou seja, não ocorreram "travamentos" no protótipo nem queda de conexão com o banco de dados. Desta forma, dados não foram perdidos durante sua coleta, o que aumenta a confiabilidade das informações obtidas. Conclui-se que os ativos de uma rede de dados podem ser gerenciados com a utilização deste protótipo.

Devido ao protótipo ser de desenvolvimento e distribuição gratuita, possui um grande diferencial quando comparado aos softwares existentes no mercado, sendo assim, pequenas e médias empresas podem utilizá-lo no gerenciamento de suas redes.

Como próximos trabalhos podem-se desenvolver novas funcionalidades, dentre elas:

- Gerenciamento gráfico
- Novos tipos de relatórios
- Gerenciamento simultâneo de grupos
- Desenvolvimento para Sistema operacional Linux

REFERÊNCIAS

CARVALHO, Tereza Cristina Melo de Brito. Gerenciamento de redes: uma abordagem de sistemas. São Paulo: Makron Books,1993

OSIER, BATSON E GROBMAN – Aprenda em 14 dias Delphi 3 – Editora Campos, 1988

TANENBAUM, Andrew S. – Redes de Computadores – Editora Campus, 1997 – 3º edição

YURI Diógenes – Certificação Cisco – CCNA 4.0 – Guia de Certificação Para o Exame #640-801 – 3º edição – Editora Axcel Books

MARTINS Agnaldo Lopes, MARANI Sandro - Topologia de Redes – 1995

Marcondes, Paulo - Apostila de TCP/IP -1996

Projeto de Redes. Disponível em: http://www.projetoderedes.com.br/tutoriais/tutorial_equipamentos_de_redes_01.php. Acesso em 29 out 2006.

CISCO SYSTEMS, Inc – Disponível em: http://www.cisco.com/en/US/products/ps6021/index.html. Acesso em 29 out 2006.

CISCO SYSTEMS, Inc – Disponível em: http://www.cisco.com/en/US/products/ps6545/index.html. Acesso em 29 out 2006.

CISCO SYSTEMS, Inc – Disponível em: http://www.cisco.com/en/US/products/ps5854/index.html. Acesso em 29 out 2006.

Projeto de Redes. Disponível em: http://www.projetoderedes.com.br/artigos/artigo_gerenciamento_de_redes_de_computadores.php. Acesso em 29 out 2006.

Projeto de Redes. Disponível em: http://www.projetoderedes.com.br/tutoriais/tutorial_equipamentos_de_redes_02.php. Acesso em 29 out 2006.

RMON. Disponível em: http://www.rnp.br/newsgen/9901/rmon.html. Acesso em 10 set 2006.

CMIP. Disponível em: http://penta.ufrgs.br/gr952/trab1/z_cmip.html. Acesso em 12 set 2006.

GERÊNCIA. Disponível em: http://www.rnp.br/newsgen/9712/gerencia.html. Acesso em 26 set 2006.

ANEXO 1 – BIBLIOTECA PING

```
{
Author:       François PIETTE
Description:  This unit encapsulate the ICMP.DLL into a VCL of type TPing.
              Using this object, you can easily ping any host on your
network.
              Works only in 32 bits mode (no Delphi 1) under NT or 95.
              If you wants to build a console mode program, use the TICMP
              object. You'll have a much smaller program.
EMail:        http://users.swing.be/francois.piette
francois.piette@swing.be
              http://www.rtfm.be/fpiette
francois.piette@rtfm.be
              francois.piette@pophost.eunet.be
Creation:     January 6, 1997
Version:      1.11
Support:      Use the mailing list twsocket@rtfm.be See website for
details.
Legal issues: Copyright (C) 1997 - 2000 by François PIETTE
              Rue de Grady 24, 4053 Embourg, Belgium. Fax: +32-4-365.74.56
              <francois.piette@pophost.eunet.be>

              This software is provided 'as-is', without any express or
              implied warranty.  In no event will the author be held liable
              for any  damages arising from the use of this software.

              Permission is granted to anyone to use this software for any
              purpose, including commercial applications, and to alter it
              and redistribute it freely, subject to the following
              restrictions:

              1. The origin of this software must not be misrepresented,
                 you must not claim that you wrote the original software.
                 If you use this software in a product, an acknowledgment
                 in the product documentation would be appreciated but is
                 not required.

              2. Altered source versions must be plainly marked as such,
and
                 must not be misrepresented as being the original software.

              3. This notice may not be removed or altered from any source
                 distribution.

              4. You must register this software by sending a picture
postcard
                 to the author. Use a nice stamp and mention your name,
street
                 address, EMail address and any comment you like to say.

Updates:
Nov 30, 1997 V1.00 Added DNSLookup capability (taken from TWSocket)
Dec 13, 1997 V1.01 Added OnEchoRequest and OnEchoReply events and removed
the
             corresponding OnDisplay event. This require to modify existing
             programs.
May 05, 1998 V1.02 Changed lpszClassName from 'XSocketWindowClass' to
             'ICSPingWindowClass' to avoid class name conflict with
TWSocket.
             Thanks to Bill Parke <econmodel@econmodel.com> who found the
```

```
            problem.
Dec 26, 1998 V1.10 Changed all events to make sender reference TPing object
            and added an argument 'Icmp' which point to the underlaying
TIcmp
            object (this was the sender in previous version). This require
            modification of existing code.
Jan 24, 1999 V1.11 Surfaced Flags property to allow fragmentation check
            (Flags = $02 to enable fragmentation check)
Nov 10, 2002 V1.12 Changed argument name from Error to Status in
OnEchoReply
            to better reflect his use. 0 means OK !

}

unit Ping;

{$IFDEF VER80}
// This source file is *NOT* compatible with Delphi 1 because it uses
// Win 32 features.
{$ENDIF}

interface

uses
  Windows, Messages, SysUtils, Classes, Winsock, Icmp;

const
  PingVersion           = 111;
  CopyRight : String    = ' TPing (c) 1997-2000 F. Piette V1.11 ';
  WM_ASYNCGETHOSTBYNAME = WM_USER + 2;

type
  TDnsLookupDone = procedure (Sender: TObject; Error: Word) of object;
  TPingDisplay   = procedure(Sender: TObject; Icmp: TObject; Msg : String)
of object;
  TPingReply     = procedure(Sender: TObject; Icmp: TObject; Status :
Integer) of object;
  TPingRequest   = procedure(Sender: TObject; Icmp: TObject) of object;
  TPing = class(TComponent)
  private
    FIcmp             : TICMP;
    FWindowHandle     : HWND;
    FDnsLookupBuffer  : array [0..MAXGETHOSTSTRUCT] of char;
    FDnsLookupHandle  : THandle;
    FDnsResult        : String;
    FOnDnsLookupDone  : TDnsLookupDone;
    FOnEchoRequest    : TPingRequest;
    FOnEchoReply      : TPingReply;
    FOnDisplay        : TPingDisplay;
  protected
    procedure   WndProc(var MsgRec: TMessage);
    procedure   WMAsyncGetHostByName(var msg: TMessage); message
WM_ASYNCGETHOSTBYNAME;
    procedure   SetAddress(Value : String);
    function    GetAddress : String;
    procedure   SetSize(Value : Integer);
    function    GetSize : Integer;
    procedure   SetTimeout(Value : Integer);
    function    GetTimeout : Integer;
    function    GetReply : TIcmpEchoReply;
    function    GetErrorCode : Integer;
```

```
    function    GetErrorString : String;
    function    GetHostName : String;
    function    GetHostIP : String;
    procedure   SetTTL(Value : Integer);
    function    GetTTL : Integer;
    procedure   Setflags(Value : Integer);
    function    Getflags : Integer;
    procedure   IcmpEchoReply(Sender: TObject; Error : Integer);
    procedure   IcmpEchoRequest(Sender: TObject);
    procedure   IcmpDisplay(Sender: TObject; Msg: String);
  public
    constructor Create(Owner : TComponent); override;
    destructor  Destroy; override;
    function    Ping : Integer;
    procedure   DnsLookup(HostName : String); virtual;
    procedure   CancelDnsLookup;

    property    Reply       : TIcmpEchoReply read GetReply;
    property    ErrorCode   : Integer        read GetErrorCode;
    property    ErrorString : String         read GetErrorString;
    property    HostName    : String         read GetHostName;
    property    HostIP      : String         read GetHostIP;
    property    Handle      : HWND           read FWindowHandle;
    property    DnsResult   : String         read FDnsResult;
  published
    property    Address     : String         read  GetAddress
                                             write SetAddress;
    property    Size        : Integer        read  GetSize
                                             write SetSize;
    property    Timeout     : Integer        read  GetTimeout
                                             write SetTimeout;
    property    TTL         : Integer        read  GetTTL
                                             write SetTTL;
    property    Flags       : Integer        read  Getflags
                                             write SetFlags;
    property    OnDisplay   : TPingDisplay   read  FOnDisplay
                                             write FOnDisplay;
    property    OnEchoRequest : TPingRequest read  FOnEchoRequest
                                             write FOnEchoRequest;
    property    OnEchoReply   : TPingReply   read  FOnEchoReply
                                             write FOnEchoReply;
    property    OnDnsLookupDone : TDnsLookupDone
                                             read  FOnDnsLookupDone
                                             write FOnDnsLookupDone;
  end;

procedure Register;

implementation


{* * * *
```

```
{ handle the message sent by windows (winsock) to handle messages.      }
function XSocketWindowProc(
    ahWnd   : HWND;
    auMsg   : Integer;
    awParam : WPARAM;
    alParam : LPARAM): Integer; stdcall;
var
    Obj    : TPing;
    MsgRec : TMessage;
begin
    { At window creation ask windows to store a pointer to our object   }
    Obj := TPing(GetWindowLong(ahWnd, 0));

    { If the pointer is not assigned, just call the default procedure   }
    if not Assigned(Obj) then
        Result := DefWindowProc(ahWnd, auMsg, awParam, alParam)
    else begin
        { Delphi use a TMessage type to pass paramter to his own kind of}
        { windows procedure. So we are doing the same...                }
        MsgRec.Msg    := auMsg;
        MsgRec.wParam := awParam;
        MsgRec.lParam := alParam;
        Obj.WndProc(MsgRec);
        Result := MsgRec.Result;
    end;
end;


{* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *}
{ This global variable is used to store the windows class characteristic}
{ and is needed to register the window class used by TWSocket           }
var
    XSocketWindowClass: TWndClass = (
        style         : 0;
        lpfnWndProc   : @XSocketWindowProc;
        cbClsExtra    : 0;
        cbWndExtra    : SizeOf(Pointer);
        hInstance     : 0;
        hIcon         : 0;
        hCursor       : 0;
        hbrBackground : 0;
        lpszMenuName  : nil;
        lpszClassName : 'ICSPingWindowClass');


{* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *}
{Allocate a window handle. This means registering a window class the first}
{time we are called, and creating a new window each time we are called.  }
function XSocketAllocateHWnd(Obj : TObject): HWND;
var
    TempClass       : TWndClass;
    ClassRegistered : Boolean;
begin
    { Check if the window class is already registered                   }
    XSocketWindowClass.hInstance := HInstance;
    ClassRegistered := GetClassInfo(HInstance,
                                    XSocketWindowClass.lpszClassName,
                                    TempClass);
    if not ClassRegistered then begin
       { Not yet registered, do it right now                            }
       Result := Windows.RegisterClass(XSocketWindowClass);
```

```
        if Result = 0 then
            Exit;
    end;

    { Now create a new window                                                    }
    Result := CreateWindowEx(WS_EX_TOOLWINDOW,
                             XSocketWindowClass.lpszClassName,
                             '',          { Window name }
                             WS_POPUP,    { Window Style}
                             0, 0,        { X, Y        }
                             0, 0,        { Width, Height }
                             0,           { hWndParent  }
                             0,           { hMenu       }
                             HInstance,   { hInstance   }
                             nil);        { CreateParam }

    { if successfull, the ask windows to store the object reference             }
    { into the reserved byte (see RegisterClass)                                }
    if (Result <> 0) and Assigned(Obj) then
        SetWindowLong(Result, 0, Integer(Obj));
end;


{* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *}
{ Free the window handle                                                        }
procedure XSocketDeallocateHWnd(Wnd: HWND);
begin
    DestroyWindow(Wnd);
end;


{* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *}
procedure TPing.WndProc(var MsgRec: TMessage);
begin
    with MsgRec do begin
        if Msg = WM_ASYNCGETHOSTBYNAME then
            WMAsyncGetHostByName(MsgRec)
        else
            Result := DefWindowProc(Handle, Msg, wParam, lParam);
    end;
end;


{* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *}
procedure TPing.WMAsyncGetHostByName(var msg: TMessage);
var
    Phe     : Phostent;
    IPAddr  : TInAddr;
    Error   : Word;
begin
    if msg.wParam <> LongInt(FDnsLookupHandle) then
        Exit;
    FDnsLookupHandle := 0;
    Error := Msg.LParamHi;
    if Error = 0 then begin
        Phe      := PHostent(@FDnsLookupBuffer);
        IPAddr   := PInAddr(Phe^.h_addr_list^)^;
        FDnsResult := StrPas(inet_ntoa(IPAddr));
    end;
    if Assigned(FOnDnsLookupDone) then
        FOnDnsLookupDone(Self, Error);
```

```
end;

{* * * *
```

```
end;

{* * * *
```

```
            Result := FIcmp.Address
        else
            Result := '';
    end;


    {* * * *
```

```
procedure TPing.SetFlags(Value : Integer);
begin
    if Assigned(FIcmp) then
        FIcmp.Flags := Value;
end;

{* * * *
```

```
        Result := FIcmp.HostIP
    else
        Result := '';
end;


{* * * *
```

```
{

Author:       François PIETTE
Description:  This unit encapsulate the ICMP.DLL into an object of type
TICMP.
              Using this object, you can easily ping any host on your
network.
              Works only in 32 bits mode (no Delphi 1) under NT or 95.
              TICMP is perfect for a console mode program, but if you build
a
              GUI program, you could use the TPing object wich is a true
VCL
              encapsulating the TICMP object. Then you can use object
inspector
              to change properties or event handler. This is much simpler
to
              use for a GUI program.
EMail:        http://users.swing.be/francois.piette
francois.piette@swing.be
              http://www.rtfm.be/fpiette
francois.piette@rtfm.be
              francois.piette@pophost.eunet.be
Creation:     January 6, 1997
Version:      1.04
Support:      Use the mailing list twsocket@rtfm.be See website for
details.
Legal issues: Copyright (C) 1997-2000 by François PIETTE
              Rue de Grady 24, 4053 Embourg, Belgium. Fax: +32-4-365.74.56
              <francois.piette@pophost.eunet.be>

              This software is provided 'as-is', without any express or
              implied warranty.  In no event will the author be held liable
              for any  damages arising from the use of this software.

              Permission is granted to anyone to use this software for any
              purpose, including commercial applications, and to alter it
              and redistribute it freely, subject to the following
              restrictions:

              1. The origin of this software must not be misrepresented,
                 you must not claim that you wrote the original software.
                 If you use this software in a product, an acknowledgment
                 in the product documentation would be appreciated but is
                 not required.

              2. Altered source versions must be plainly marked as such,
and
                 must not be misrepresented as being the original software.

              3. This notice may not be removed or altered from any source
                 distribution.

              4. You must register this software by sending a picture
postcard
                 to the author. Use a nice stamp and mention your name,
street
                 address, EMail address and any comment you like to say.

Updates:
```

```
Dec 13, 1997 V1.01 Added OnEchoRequest and OnEchoReply events and removed
the
             corresponding OnDisplay event. This require to modify existing
             programs.
Mar 15, 1998 V1.02 Deplaced address resolution just before use
Sep 24, 1998 V1.93 Changed TIPAddr and others to LongInt to avoid range
error
             problems with Delphi 4
Jan 24, 1999 V1.11 Surfaced Flags property to allow fragmentation check
             (Flags = IP_FLAG_DF to enable fragmentation check)

}

unit Icmp;

interface

{$IFDEF VER80}
// This source file is *NOT* compatible with Delphi 1 because it uses
// Win 32 features.
{$ENDIF}

uses
  Windows, SysUtils, Classes, WinSock;

const
  IcmpVersion = 102;
  IcmpDLL     = 'icmp.dll';

  // IP status codes returned to transports and user IOCTLs.
  IP_SUCCESS                = 0;
  IP_STATUS_BASE            = 11000;
  IP_BUF_TOO_SMALL          = (IP_STATUS_BASE + 1);
  IP_DEST_NET_UNREACHABLE   = (IP_STATUS_BASE + 2);
  IP_DEST_HOST_UNREACHABLE  = (IP_STATUS_BASE + 3);
  IP_DEST_PROT_UNREACHABLE  = (IP_STATUS_BASE + 4);
  IP_DEST_PORT_UNREACHABLE  = (IP_STATUS_BASE + 5);
  IP_NO_RESOURCES           = (IP_STATUS_BASE + 6);
  IP_BAD_OPTION             = (IP_STATUS_BASE + 7);
  IP_HW_ERROR               = (IP_STATUS_BASE + 8);
  IP_PACKET_TOO_BIG         = (IP_STATUS_BASE + 9);
  IP_REQ_TIMED_OUT          = (IP_STATUS_BASE + 10);
  IP_BAD_REQ                = (IP_STATUS_BASE + 11);
  IP_BAD_ROUTE              = (IP_STATUS_BASE + 12);
  IP_TTL_EXPIRED_TRANSIT    = (IP_STATUS_BASE + 13);
  IP_TTL_EXPIRED_REASSEM    = (IP_STATUS_BASE + 14);
  IP_PARAM_PROBLEM          = (IP_STATUS_BASE + 15);
  IP_SOURCE_QUENCH          = (IP_STATUS_BASE + 16);
  IP_OPTION_TOO_BIG         = (IP_STATUS_BASE + 17);
  IP_BAD_DESTINATION        = (IP_STATUS_BASE + 18);

  // status codes passed up on status indications.
  IP_ADDR_DELETED           = (IP_STATUS_BASE + 19);
  IP_SPEC_MTU_CHANGE        = (IP_STATUS_BASE + 20);
  IP_MTU_CHANGE             = (IP_STATUS_BASE + 21);

  IP_GENERAL_FAILURE        = (IP_STATUS_BASE + 50);

  MAX_IP_STATUS             = IP_GENERAL_FAILURE;

  IP_PENDING                = (IP_STATUS_BASE + 255);
```

```
  // IP header flags
  IP_FLAG_DF                  = $02;        // Don't fragment this packet.

  // IP Option Types
  IP_OPT_EOL                  = $00;        // End of list option
  IP_OPT_NOP                  = $01;        // No operation
  IP_OPT_SECURITY             = $82;        // Security option.
  IP_OPT_LSRR                 = $83;        // Loose source route.
  IP_OPT_SSRR                 = $89;        // Strict source route.
  IP_OPT_RR                   = $07;        // Record route.
  IP_OPT_TS                   = $44;        // Timestamp.
  IP_OPT_SID                  = $88;        // Stream ID (obsolete)
  MAX_OPT_SIZE                = $40;

type
  // IP types
  TIPAddr   = LongInt;   // An IP address.
  TIPMask   = LongInt;   // An IP subnet mask.
  TIPStatus = LongInt;   // Status code returned from IP APIs.

  PIPOptionInformation = ^TIPOptionInformation;
  TIPOptionInformation = packed record
     TTL:         Byte;      // Time To Live (used for traceroute)
     TOS:         Byte;      // Type Of Service (usually 0)
     Flags:       Byte;      // IP header flags (usually 0)
     OptionsSize: Byte;      // Size of options data (usually 0, max 40)
     OptionsData: PChar;     // Options data buffer
  end;

  PIcmpEchoReply = ^TIcmpEchoReply;
  TIcmpEchoReply = packed record
     Address:       TIPAddr;              // Replying address
     Status:        DWord;                // IP status value
     RTT:           DWord;                // Round Trip Time in
milliseconds
     DataSize:      Word;                 // Reply data size
     Reserved:      Word;                 // Reserved
     Data:          Pointer;              // Pointer to reply data buffer
     Options:       TIPOptionInformation; // Reply options
  end;

  // IcmpCreateFile:
  //     Opens a handle on which ICMP Echo Requests can be issued.
  // Arguments:
  //     None.
  // Return Value:
  //     An open file handle or INVALID_HANDLE_VALUE. Extended error
information
  //     is available by calling GetLastError().
  TIcmpCreateFile  = function: THandle; stdcall;

  // IcmpCloseHandle:
  //     Closes a handle opened by ICMPOpenFile.
  // Arguments:
  //     IcmpHandle  - The handle to close.
  // Return Value:
  //     TRUE if the handle was closed successfully, otherwise FALSE.
Extended
  //     error information is available by calling GetLastError().
  TIcmpCloseHandle = function(IcmpHandle: THandle): Boolean; stdcall;
```

```
  // IcmpSendEcho:
  //     Sends an ICMP Echo request and returns one or more replies. The
  //     call returns when the timeout has expired or the reply buffer
  //     is filled.
  // Arguments:
  //     IcmpHandle         - An open handle returned by ICMPCreateFile.
  //     DestinationAddress - The destination of the echo request.
  //     RequestData        - A buffer containing the data to send in the
  //                          request.
  //     RequestSize        - The number of bytes in the request data
buffer.
  //     RequestOptions     - Pointer to the IP header options for the
request.
  //                          May be NULL.
  //     ReplyBuffer        - A buffer to hold any replies to the request.
  //                          On return, the buffer will contain an array
of
  //                          ICMP_ECHO_REPLY structures followed by
options
  //                          and data. The buffer should be large enough
to
  //                          hold at least one ICMP_ECHO_REPLY structure
  //                          and 8 bytes of data - this is the size of
  //                          an ICMP error message.
  //     ReplySize          - The size in bytes of the reply buffer.
  //     Timeout            - The time in milliseconds to wait for replies.
  // Return Value:
  //     Returns the number of replies received and stored in ReplyBuffer.
If
  //     the return value is zero, extended error information is available
  //     via GetLastError().
  TIcmpSendEcho     = function(IcmpHandle:          THandle;
                               DestinationAddress:  TIPAddr;
                               RequestData:         Pointer;
                               RequestSize:         Word;
                               RequestOptions:      PIPOptionInformation;
                               ReplyBuffer:         Pointer;
                               ReplySize:           DWord;
                               Timeout:             DWord
                              ): DWord; stdcall;

  // Event handler type declaration for TICMP.OnDisplay event.
  TICMPDisplay = procedure(Sender: TObject; Msg : String) of object;
  TICMPReply   = procedure(Sender: TObject; Error : Integer) of object;

  // The object wich encapsulate the ICMP.DLL
  TICMP = class(TObject)
  private
    hICMPdll :          HModule;                    // Handle for ICMP.DLL
    IcmpCreateFile :    TIcmpCreateFile;
    IcmpCloseHandle :   TIcmpCloseHandle;
    IcmpSendEcho :      TIcmpSendEcho;
    hICMP :             THandle;                    // Handle for the ICMP
Calls
    FReply :            TIcmpEchoReply;             // ICMP Echo reply buffer
    FAddress :          String;                     // Address given
    FHostName :         String;                     // Dotted IP of host
(output)
    FHostIP :           String;                     // Name of host
(Output)
```

```
    FIPAddress :      TIPAddr;                    // Address of host to
contact
    FSize :           Integer;                    // Packet size (default
to 56)
    FTimeOut :        Integer;                    // Timeout (default to
4000mS)
    FTTL :            Integer;                    // Time To Live (for
send)
    FFlags :          Integer;                    // Options flags
    FOnDisplay :      TICMPDisplay;               // Event handler to
display
    FOnEchoRequest :  TNotifyEvent;
    FOnEchoReply :    TICMPReply;
    FLastError :      DWORD;                      // After sending ICMP
packet
    FAddrResolved :   Boolean;
    procedure ResolveAddr;
  public
    constructor Create; virtual;
    destructor  Destroy; override;
    function    Ping : Integer;
    procedure   SetAddress(Value : String);
    function    GetErrorString : String;

    property Address        : String          read  FAddress     write
SetAddress;
    property Size           : Integer         read  FSize        write FSize;
    property Timeout        : Integer         read  FTimeout     write
FTimeout;
    property Reply          : TIcmpEchoReply  read  FReply;
    property TTL            : Integer         read  FTTL         write FTTL;
    Property Flags          : Integer         read  FFlags       write FFlags;
    property ErrorCode      : DWORD           read  FLastError;
    property ErrorString    : String          read  GetErrorString;
    property HostName       : String          read  FHostName;
    property HostIP         : String          read  FHostIP;
    property OnDisplay      : TICMPDisplay    read  FOnDisplay write
FOnDisplay;
    property OnEchoRequest  : TNotifyEvent    read  FOnEchoRequest
                                              write FOnEchoRequest;
    property OnEchoReply    : TICMPReply      read  FOnEchoReply
                                              write FOnEchoReply;
  end;

  TICMPException = class(Exception);

implementation

{* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *}
constructor TICMP.Create;
var
    WSAData: TWSAData;
begin
    hICMP     := INVALID_HANDLE_VALUE;
    FSize     := 56;
    FTTL      := 64;
    FTimeOut  := 4000;

    // initialise winsock
    if WSAStartup($101, WSAData) <> 0 then
        raise TICMPException.Create('Error initialising Winsock');
```

```
    // register the icmp.dll stuff
    hICMPdll := LoadLibrary(icmpDLL);
    if hICMPdll = 0 then
        raise TICMPException.Create('Unable to register ' + icmpDLL);

    @ICMPCreateFile  := GetProcAddress(hICMPdll, 'IcmpCreateFile');
    @IcmpCloseHandle := GetProcAddress(hICMPdll, 'IcmpCloseHandle');
    @IcmpSendEcho    := GetProcAddress(hICMPdll, 'IcmpSendEcho');

    if (@ICMPCreateFile = Nil) or
       (@IcmpCloseHandle = Nil) or
       (@IcmpSendEcho = Nil) then
          raise TICMPException.Create('Error loading dll functions');

    hICMP := IcmpCreateFile;
    if hICMP = INVALID_HANDLE_VALUE then
        raise TICMPException.Create('Unable to get ping handle');
end;


{* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *}
destructor TICMP.Destroy;
begin
    if hICMP <> INVALID_HANDLE_VALUE then
        IcmpCloseHandle(hICMP);
    if hICMPdll <> 0 then
        FreeLibrary(hICMPdll);
    WSACleanup;
    inherited Destroy;
end;


{* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *}
function MinInteger(X, Y: Integer): Integer;
begin
    if X >= Y then
        Result := Y
    else
        Result := X;
end;


{* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *}
procedure TICMP.ResolveAddr;
var
    Phe : PHostEnt;              // HostEntry buffer for name lookup
begin
    // Convert host address to IP address
    FIPAddress := inet_addr(PChar(FAddress));
    if FIPAddress <> LongInt(INADDR_NONE) then
        // Was a numeric dotted address let it in this format
        FHostName := FAddress
    else begin
        // Not a numeric dotted address, try to resolve by name
        Phe := GetHostByName(PChar(FAddress));
        if Phe = nil then begin
            FLastError := GetLastError;
            if Assigned(FOnDisplay) then
                FOnDisplay(Self, 'Unable to resolve ' + FAddress);
            Exit;
```

```
        end;

        FIPAddress := longint(plongint(Phe^.h_addr_list^)^);
        FHostName  := Phe^.h_name;
    end;

    FHostIP       := StrPas(inet_ntoa(TInAddr(FIPAddress)));
    FAddrResolved := TRUE;
end;


{* * * *
```

```
    pReqData, pData:    Pointer;
    pIPE:               PIcmpEchoReply;       // ICMP Echo reply buffer
    IPOpt:              TIPOptionInformation; // IP Options for packet to send
    Msg:                String;
  begin
    Result     := 0;
    FLastError := 0;

    if not FAddrResolved then
        ResolveAddr;

    if FIPAddress = LongInt(INADDR_NONE) then begin
        FLastError := IP_BAD_DESTINATION;
        if Assigned(FOnDisplay) then
            FOnDisplay(Self, 'Invalid host address');
        Exit;
    end;

    // Allocate space for data buffer space
    BufferSize := SizeOf(TICMPEchoReply) + FSize;
    GetMem(pReqData, FSize);
    GetMem(pData,    FSize);
    GetMem(pIPE,     BufferSize);

    try
        // Fill data buffer with some data bytes
        FillChar(pReqData^, FSize, $20);
        Msg := 'Pinging from Delphi code written by F. Piette';
        Move(Msg[1], pReqData^, MinInteger(FSize, Length(Msg)));

        pIPE^.Data := pData;
        FillChar(pIPE^, SizeOf(pIPE^), 0);

        if Assigned(FOnEchoRequest) then
            FOnEchoRequest(Self);

        FillChar(IPOpt, SizeOf(IPOpt), 0);
        IPOpt.TTL    := FTTL;
        IPOpt.Flags  := FFlags;
        Result       := IcmpSendEcho(hICMP, FIPAddress, pReqData, FSize,
                                     @IPOpt, pIPE, BufferSize, FTimeOut);
        FLastError   := GetLastError;
        FReply       := pIPE^;

        if Assigned(FOnEchoReply) then
            FOnEchoReply(Self, Result);
    finally
        // Free those buffers
        FreeMem(pIPE);
        FreeMem(pData);
        FreeMem(pReqData);
    end;
end;

{* * * *
```

```
unit Form_Banco;

interface

uses
  SysUtils, Classes, DB, Provider, DBXpress, SqlExpr, FMTBcd,
DBClient,
  IBDatabase, IBCustomDataSet, IBTable, IBQuery, Dialogs;
type
  TBanco = class(TDataModule)
    Conectar: TSQLConnection;
    DSGrupo: TDataSource;
    tb_grupo: TIBTable;
    Banco: TIBDatabase;
    Transaction: TIBTransaction;
    DSAtivosRede: TDataSource;
    tb_AtivoRede: TIBTable;
    tb_Historico: TIBTable;
    DSHistorico: TDataSource;
    DSQuery: TDataSource;
    Query: TIBQuery;
    Query1: TIBQuery;
    procedure tb_grupoPostError(DataSet: TDataSet; E:
EDatabaseError;
      var Action: TDataAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Banco: TBanco;
implementation
{$R *.dfm}

// rotina para tratamento de erros amigaveis quando faz insercao de
dados
procedure TBanco.tb_grupoPostError(DataSet: TDataSet; E:
EDatabaseError;
  var Action: TDataAction);
var
erroMens : string;
begin
ErroMens := E.Message;
if ErroMens = 'Key violation.' then // testa se se já existe a chave
primária
  begin
  ShowMessage('Chave primária inválida !');
  action := daAbort;
  end;
if ErroMens <> '' then
  begin
  ShowMessage('Erro de inserção !');
  action := daAbort;
  end;
end;

end.
```

APÊNDICE 2 - CÓDIGO FONTE - Form_CadastroComputadores

```
unit Form_CadastroComputadores;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, DBCtrls, Mask, DB, ExtCtrls, DBTables, Grids,
DBGrids,
  ComCtrls, Buttons, Form_Banco;
type
  TFormCadastroComputadores = class(TForm)
    Bevel1: TBevel;
    StatusBar1: TStatusBar;
    GroupBox1: TGroupBox;
    DBNavigator1: TDBNavigator;
    Label1: TLabel;
    DBEdit1: TDBEdit;
    Label2: TLabel;
    DBEdit4: TDBEdit;
    Label4: TLabel;
    Label3: TLabel;
    DBEdit3: TDBEdit;
    DBMemo1: TDBMemo;
    Label5: TLabel;
    GroupBox2: TGroupBox;
    DBGrid1: TDBGrid;
    DBEdit2: TDBEdit;
    ComboBox1: TComboBox;
    Edit1: TEdit;
    Button1: TButton;
    procedure ComboBox1Change(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure DBGrid1CellClick(Column: TColumn);
    procedure  DBGrid1DrawColumnCell(Sender:  TObject;  const  Rect:
TRect;
      DataCol: Integer; Column: TColumn; State: TGridDrawState);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  FormCadastroComputadores: TFormCadastroComputadores;
implementation
{$R *.dfm}
procedure          TFormCadastroComputadores.ComboBox1Change(Sender:
TObject);
begin
   // Traz todos os grupos e mostra a descricao do grupo
    Banco.tb_AtivoRede.Edit;
    Banco.Query.Close;
    Banco.Query.DataSource  :=  Banco.DSGrupo;  //  escolhe  a  tabela
Tb_grupos
    Banco.Query.Open;
    Banco.Query.SQL.Clear;
    Banco.Query.SQL.Add('SELECT     *     FROM     tb_grupos     WHERE
cod_grupo='+ComboBox1.Text+'');
```

```
          Banco.Query.Open;
          Edit1.Text := Banco.Query.Fields[1].AsString;
          DBEdit2.Field.Text := ComboBox1.Text;
       end;

       procedure TFormCadastroComputadores.FormActivate(Sender: TObject);
       begin
       // quanto ativa o formulario traz todos os grupos cadastrados no
componente combobox
          ComboBox1.Items.Clear;
          Banco.tb_grupo.First;
          while not Banco.tb_Grupo.Eof do
          begin

ComboBox1.Items.Add(Banco.tb_Grupo.FieldByName('Cod_grupo').AsString);
             Banco.tb_Grupo.Next;
          end;
       end;
       procedure TFormCadastroComputadores.FormClose(Sender: TObject;
         var Action: TCloseAction);
       begin
          DBEdit1.SetFocus;       // forca o focus no campo dbedit1
          Banco.tb_grupo.Cancel;    // cancela as opecacoes pendentes se
houver
       end;
       procedure          TFormCadastroComputadores.DBGrid1CellClick(Column:
TColumn);
       begin
         // quando clicar no dbgrid traz no componente combobox o conteudo
do grupo
         combobox1.Text := dbgrid1.Columns.Grid.Fields[1].AsString;
       end;
       procedure     TFormCadastroComputadores.DBGrid1DrawColumnCell(Sender:
TObject;
         const Rect: TRect; DataCol: Integer; Column: TColumn;
         State: TGridDrawState);
       begin
          // quando clicar no dbgrid traz no componente combobox o conteudo
do grupo
          combobox1.Text := dbgrid1.Columns.Grid.Fields[1].AsString;
       end;
       procedure TFormCadastroComputadores.Button1Click(Sender: TObject);
       begin
       // pergunta se deseja realmente excluir o item selecionado
       if messagedlg('Confirma a Exclusão ?',mtwarning,[mbcancel,mbOK],0) =
MrOK then
          begin
          // caso a escolha seja OK, será excluido o ativo de rede
          banco.Query.Close;
          banco.Query.DataSource := banco.DSAtivosRede;
          banco.Query.Open;
          banco.Query.SQL.Clear;
          banco.Query.SQL.Add('DELETE     FROM     tb_computadores     where
cod_computador = '+ dbedit1.Text);
          banco.Query.Open;
          banco.tb_AtivoRede.Refresh;
          end;
       end;
       end.
```

APÊNDICE 3 - CÓDIGO FONTE - Form_CadGrupos

```
unit Form_CadGrupos;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, Mask, DBCtrls, DB, DBTables, Grids, DBGrids,
ExtCtrls, Form_Banco,
  ComCtrls, Buttons;

type
  TFormCadGrupos = class(TForm)
    StatusBar1: TStatusBar;
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    DBNavigator1: TDBNavigator;
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    DBGrid1: TDBGrid;
    Button1: TButton;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure DBNavigator1Click(Sender: TObject; Button:
TNavigateBtn);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormCadGrupos: TFormCadGrupos;

implementation

uses PingTst1;

{$R *.dfm}

procedure TFormCadGrupos.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  // quando fechar o formulario grupos coloca os grupos cadastrados
no combobox
  PingTstForm.ComboBox1.Items.Clear;
  Banco.tb_grupo.First;
  while not Banco.tb_Grupo.Eof do
  begin

PingTstForm.ComboBox1.Items.Add(Banco.tb_Grupo.FieldByName('Cod_grupo').AsS
tring);
    Banco.tb_Grupo.Next;
```

```
        end;
      end;

      procedure TFormCadGrupos.DBNavigator1Click(Sender: TObject;
        Button: TNavigateBtn);
      begin
      dbedit1.SetFocus;
      end;

      procedure TFormCadGrupos.Button1Click(Sender: TObject);
      var
      Sucesso : integer;
      begin
      // soma quantos ativos de rede existem neste grupo
      Sucesso := 0;
      banco.Query.Close;
      banco.Query.DataSource := banco.DSAtivosRede;
      banco.Query.Open;
      banco.Query.SQL.Clear;
      banco.Query.SQL.Add('SELECT COUNT(*) AS Sucesso FROM tb_computadores
where cod_grupo = ' + dbedit1.Text);
      banco.Query.Open;
      Sucesso := banco.Query.Fields[0].AsInteger;
      if Sucesso > 0 then   // caso exista ativos de rede no grupo
        begin
        ShowMessage('Existem Ativos de redes cadastrados !!!');
        end
      else                  // nao existe ativos de rede neste grupo
        begin
        if          messagedlg('Confirma          a          Exclusão
?',mtwarning,[mbcancel,mbOK],0) = MrOK then
          begin
          // caso a escolha se OK, excluira o grupo
          banco.Query.Close;
          banco.Query.DataSource := banco.DSAtivosRede;
          banco.Query.Open;
          banco.Query.SQL.Clear;
          banco.Query.SQL.Add('DELETE FROM tb_grupos where cod_grupo =
'+ dbedit1.Text);
          banco.Query.Open;
          banco.tb_grupo.Refresh;
          end;
        end;
      dbedit1.SetFocus;
      end;
      end.
```

APÊNDICE 4 - CÓDIGO FONTE - Form_GeraRelatorio

```
unit Form_GeraRelatorio;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, DB, DBTables, QRCtrls, QuickRpt, ExtCtrls, Form_Banco;

type
  TFormGeraRelatorio = class(TForm)
    QuickRep1: TQuickRep;
    QRBand1: TQRBand;
    QRSubDetail1: TQRSubDetail;
    QRLabel1: TQRLabel;
    QRLabel2: TQRLabel;
    QRLabel3: TQRLabel;
    QRLabel4: TQRLabel;
    QRDBText1: TQRDBText;
    QRDBText2: TQRDBText;
    QRDBText3: TQRDBText;
    QRDBText4: TQRDBText;
    QRLabel5: TQRLabel;
    QRPerdas: TQRLabel;
    QRLabel7: TQRLabel;
    QRTotal: TQRLabel;
    QRLabel9: TQRLabel;
    QRSucesso: TQRLabel;
    QRTotalP: TQRLabel;
    QRSucessoP: TQRLabel;
    QRPerdasP: TQRLabel;
    QRNome: TQRLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormGeraRelatorio: TFormGeraRelatorio;

implementation

{$R *.dfm}

end.
```

APÊNDICE 5 - CÓDIGO FONTE - Form_RelatorioAtivo

```
unit Form_RelatorioAtivo;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, Grids, DBGrids;

type
  TFormRelatorioAtivo = class(TForm)
    GroupBox1: TGroupBox;
    BtnImprimir: TButton;
    Edit1: TEdit;
    procedure BtnImprimirClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormRelatorioAtivo: TFormRelatorioAtivo;

implementation

uses Form_GeraRelatorio, Form_Banco;

{$R *.dfm}

procedure TFormRelatorioAtivo.BtnImprimirClick(Sender: TObject);
var
Sucesso, Total, PorcentualP, Perdas, PorcentualS : Real;
begin
    if edit1.Text = '' then  // caso enao escolha nada
       begin
       showmessage('É necessario informar o IP!');
       edit1.SetFocus;
       end
    else
       begin
       // soma na variavel SUCESSO a quantidade de vezes o ativo escolhido
alcancou o destino
       Banco.Query.Close;
       Banco.Query.DataSource := Banco.DSHistorico;
       Banco.Query.Open;
       Banco.Query.SQL.Clear;
       Banco.Query.SQL.Add('SELECT COUNT(*) AS Sucesso FROM tb_historico
WHERE Host = "'+Edit1.Text+'" AND Resposta <> 4194324');
       Banco.Query.Open;
       Sucesso := Banco.Query.Fields[0].AsInteger;

       // soma na variavel TOTAL a quantidade de vezes o ativo escolhido
foi testado
       Banco.Query.Close;
       Banco.Query.DataSource := Banco.DSHistorico;
```

```
        Banco.Query.Open;
        Banco.Query.SQL.Clear;
        Banco.Query.SQL.Add('SELECT COUNT(*) AS Total FROM tb_historico
 WHERE Host = "'+Edit1.Text+'"');
        Banco.Query.Open;
        Total := Banco.Query.Fields[0].AsInteger;

        // seleciona to o historico do ativo escolhido
        Banco.Query.Close;
        Banco.Query.DataSource := Banco.DSHistorico;
        Banco.Query.Open;
        Banco.Query.SQL.Clear;
        Banco.Query.SQL.Add('SELECT * FROM tb_historico WHERE Host =
"'+Edit1.Text+'"');
        Banco.Query.Open;

        if total > 0 then
           begin
           // faz o percentual
           Perdas      := Total - Sucesso;
           PorcentualP := Perdas * 100 / Total;
           PorcentualS := Sucesso * 100 / Total;
           end;

        // faz a formatação necessária para a apresentacao do totais e os
percentuais
        FormGeraRelatorio.QRPerdasP.Caption :=
Format('%3n',[PorcentualP])+'%';
        FormGeraRelatorio.QRSucessoP.Caption :=
Format('%3n',[PorcentualS])+'%';
        FormGeraRelatorio.QRTotalP.Caption := '100%';

        FormGeraRelatorio.QRPerdas.Caption := Format('%5n',[Perdas]);
        FormGeraRelatorio.QRSucesso.Caption := Format('%5n',[Sucesso]);
        FormGeraRelatorio.QRTotal.Caption := Format('%5n',[Total]);
        FormGeraRelatorio.QRNome.caption := ' Relatório de Disponibilidade
do Ativo de Rede ';

        FormGeraRelatorio.QuickRep1.Preview;
    end;
end;

end.
```

APÊNDICE 6 - CÓDIGO FONTE - Form_RelatorioPeriodo

```
unit Form_RelatorioPeriodo;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, Buttons, StdCtrls, ExtCtrls, ComCtrls, Grids, DBGrids, DB,
  DBTables, Form_Banco;

type
  TFormRelatorioPeriodo = class(TForm)
    Bevel1: TBevel;
    StatusBar1: TStatusBar;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    Label1: TLabel;
    DateTimePicker1: TDateTimePicker;
    Label2: TLabel;
    DateTimePicker2: TDateTimePicker;
    BtnImprimir: TSpeedButton;
    Bevel2: TBevel;
    BtnApagar: TSpeedButton;
    procedure BtnImprimirClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure BtnApagarClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormRelatorioPeriodo: TFormRelatorioPeriodo;

implementation

uses Form_GeraRelatorio;

{$R *.dfm}

procedure TFormRelatorioPeriodo.BtnImprimirClick(Sender: TObject);
var
  Inicio, Final : string;
  Sucesso, Total, PorcentualP, Perdas, PorcentualS : Real;
begin
// faz verificacao entre a data inicial e a data final
If DateTimePicker2.Date < DateTimePicker1.Date Then
   begin
   ShowMessage('Intervalo de datas inválido, a data inicial é maior que a
data final!');
   DateTimePicker2.Date := DateTimePicker1.Date;
   end
else
   begin
   // soma na variavelque SUCESSO a quantidade de vezes teve sucesso nos
testes os ativos de redes no periodo escolhido
   Inicio := DateToStr(DateTimePicker1.Date);
```

```
   Final  := DateToStr(DateTimePicker2.Date);
   Banco.Query.Close;
   Banco.Query.DataSource := Banco.DSHistorico;
   Banco.Query.Open;
   Banco.Query.SQL.Clear;
   Banco.Query.SQL.Add('SELECT COUNT(*) AS Sucesso FROM tb_historico WHERE
Data >= '+#34+Inicio+#34+'AND Data <='+#34+Final+#34+' AND Resposta <>
4194324');
   Banco.Query.Open;
   Sucesso := Banco.Query.Fields[0].AsInteger;

   // soma na variavelque TOTAL a quantidade de vezes que os ativos de
redes foram testados no periodo escolhido
   Banco.Query.Close;
   Banco.Query.DataSource := Banco.DSHistorico;
   Banco.Query.Open;
   Banco.Query.SQL.Clear;
   Banco.Query.SQL.Add('SELECT COUNT(*) AS Total FROM tb_historico WHERE
Data >= '+#34+Inicio+#34+'AND Data <='+#34+Final+#34+'');
   Banco.Query.Open;
   Total := Banco.Query.Fields[0].AsInteger;

   // seleciona  todos ativos de redes que foram testados no periodo
escolhido
   Banco.Query.Close;
   Banco.Query.DataSource := Banco.DSHistorico;
   Banco.Query.Open;
   Banco.Query.SQL.Clear;
   Banco.Query.SQL.Add('SELECT * FROM tb_historico WHERE Data >=
'+#34+Inicio+#34+'AND Data <='+#34+Final+#34+'');
   Banco.Query.Open;
   Banco.Query.DataSource := Banco.DSHistorico;

   if total > 0 then
      begin
      // faz o percentual
      Perdas := Total - Sucesso;
      PorcentualP := Perdas * 100 / Total;
      PorcentualS := Sucesso * 100 / Total;
      end;

   // faz a formatacao necessaria para a apresentacao dos totais e
perncentuais
   FormGeraRelatorio.QRPerdasP.Caption := Format('%3n',[PorcentualP])+'%';
   FormGeraRelatorio.QRSucessoP.Caption := Format('%3n',[PorcentualS])+'%';
   FormGeraRelatorio.QRTotalP.Caption := '100%';

   FormGeraRelatorio.QRPerdas.Caption := Format('%5.0n',[Perdas]);
   FormGeraRelatorio.QRSucesso.Caption := Format('%5.0n',[Sucesso]);
   FormGeraRelatorio.QRTotal.Caption := Format('%5.0n',[Total]);
   FormGeraRelatorio.QRNome.caption := ' Disponibilidade dos Ativos de Rede
de ' + inicio + ' a ' + Final;
   FormGeraRelatorio.QuickRep1.Preview;
   end;
end;

procedure TFormRelatorioPeriodo.FormActivate(Sender: TObject);
begin
  // inicializa o inicio e o fim com a data do dia
  DateTimePicker1.Date := Date();
  DateTimePicker2.Date := Date();
```

```
end;

procedure TFormRelatorioPeriodo.BtnApagarClick(Sender: TObject);
var
  Inicio : string;
  Final  : string;
begin
If DateTimePicker2.Date < DateTimePicker1.Date Then
   begin
      ShowMessage('Intervalo de datas inválido, a data inicial é maior que
a data final!');
      DateTimePicker2.Date := DateTimePicker1.Date;
   end
else
   begin
   // elimina todo o historico entre as datas selecionadas
   Inicio := DateToStr(DateTimePicker1.Date);
   Final  := DateToStr(DateTimePicker2.Date);
   Banco.Query.DataSource := Banco.DSHistorico;
   Banco.Query.Close;
   Banco.Query.SQL.Clear;
   Banco.Query.SQL.Add('DELETE    FROM    tb_historico    WHERE    Data    >=
'+#34+Inicio+#34+'AND Data <='+#34+Final+#34+'');
   Banco.Query.ExecSQL;
   ShowMessage('Dados apagados com sucesso.');
  end;
end;

end.
```

APÊNDICE 7 - CÓDIGO FONTE – BIBLIOTECA PING MODIFICADA

```
{
Creditos: Biblioteca Ping
Author:       François PIETTE
Creation:     November 30, 1997
Version:      1.03
EMail:        francois.piette@overbyte.be  http://www.overbyte.be
              francois.piette@rtfm.be      http://www.rtfm.be/fpiette
              francois.piette@pophost.eunet.be

Description:  This demo show how to use the TPing object to ping any host.
Legal issues: Copyright (C) 1997-2002 by François PIETTE
              Rue de Grady 24, 4053 Embourg, Belgium. Fax: +32-4-365.74.56
              <francois.piette@overbyte.be>

Updates:
Dec 13, 1997 V1.01 Use the new OnEchoRequest and OnEchoReply events.
Dec 26, 1998 V1.02 Changed event handler for new TPing version (1.10)
Nov 10, 2002 V1.03 Added Reply.Status in display when failed
                   Changed argument name from Error to Status in EchoReply
                   event (same change has in component).

Projeto de Graduação do Curso de Tecnologia em Informatica
Faculdade Tuiuti
Orientador: Roberto Néia Amaral

Desenvolvimento do Protótipo:
Helio Ishii Stroparo
Rafael Avila Leite

Ultima Atualização: 10/11/2006
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *}
unit PingTst1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Ping, StdCtrls, Menus, DB, DBTables, ExtCtrls, ComCtrls, Grids, DBGrids,
DBIProcs, Form_Banco;

type
  TPingTstForm = class(TForm)
    Ping1: TPing;
    MainMenu1: TMainMenu;
    Cadastro1: TMenuItem;
    N1: TMenuItem;
    Grupos1: TMenuItem;
    Ferramentas1: TMenuItem;
    Histricos1: TMenuItem;
    Porgrupo1: TMenuItem;
    N4: TMenuItem;
```

```
    Perodo1: TMenuItem;
    N3: TMenuItem;
    sair2: TMenuItem;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    Bevel1: TBevel;
    Bevel2: TBevel;
    StatusBar1: TStatusBar;
    TimerAtivo: TTimer;
    TimerGrupo: TTimer;
    GroupBox1: TGroupBox;
    HostEdit: TEdit;
    PingButton: TButton;
    CancelButton: TButton;
    GroupBox3: TGroupBox;
    ComboBox1: TComboBox;
    GroupBox5: TGroupBox;
    DysplayMemoGrupo: TMemo;
    PingButtonGrupo: TButton;
    CancelButtonGrupo: TButton;
    GroupBox2: TGroupBox;
    DisplayMemo: TMemo;
    procedure PingButtonClick(Sender: TObject);
    procedure Ping1Display(Sender: TObject; Icmp: TObject; Msg: String);
    procedure Ping1DnsLookupDone(Sender: TObject; Error: Word);
    procedure CancelButtonClick(Sender: TObject);
    procedure Ping1EchoReply(Sender: TObject; Icmp: TObject; Status:
Integer);
    procedure N1Click(Sender: TObject);
    procedure Grupos1Click(Sender: TObject);
    procedure Sair1Click(Sender: TObject);
    procedure sair2Click(Sender: TObject);
    procedure Perodo1Click(Sender: TObject);
    procedure TimerAtivoTimer(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure PingButtonGrupoClick(Sender: TObject);
    procedure CancelButtonGrupoClick(Sender: TObject);
    procedure TimerGrupoTimer(Sender: TObject);
    procedure Porgrupo1Click(Sender: TObject);
    procedure PageControl1Change(Sender: TObject);
    procedure ComboBox1DropDown(Sender: TObject);
  end;

var
  PingTstForm: TPingTstForm;

implementation

uses Form_CadastroComputadores, Form_CadGrupos, Form_RelatorioPeriodo,
  Form_RelatorioAtivo;

{$R *.DFM}

{* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* *}
procedure TPingTstForm.PingButtonClick(Sender: TObject);
begin
// quando clicado no botao TESTAR
if HostEdit.Text <> '' then
   begin
```

```
   PingButton.Enabled   := false; // desabilita o botao TESTAR
   CancelButton.Enabled := true;  // habilita o botao PARAR
   TimerAtivo.Enabled   := true;  // executa o componente timer INDIVIDUAL
   end;
end;


{* * * *
```

```
{* * * *
```

```
  Banco.Query.Close;
  Banco.Query.SQL.Clear;
  Banco.Query.SQL.Add('SELECT * FROM tb_computadores WHERE
cod_grupo='+ComboBox1.Text+'');
  Banco.Query.Open;
end;

procedure TPingTstForm.PingButtonGrupoClick(Sender: TObject);
begin
if combobox1.Text <> '' then
  begin
  PingButtonGrupo.Enabled   := false;   // desabilita o botao TESTAR
  CancelButtonGrupo.Enabled := true;    // habilita o botao PARAR
  TimerGrupo.Enabled        := true;    // executa o componente TIMERGRUPO
  end;
end;

procedure TPingTstForm.CancelButtonGrupoClick(Sender: TObject);
begin
  PingButtonGrupo.Enabled   := true;    // habilita o botao TESTAR
  CancelButtonGrupo.Enabled := false;   // desabilita o botao PARAR
  TimerGrupo.Enabled        := false;   // executa o componente TIMERGRUPO
end;

procedure TPingTstForm.TimerGrupoTimer(Sender: TObject);
var
IP   : String;
begin
   // Traz todos os ativos de rede que forem do grupo escolhido através do
   // conteudo do campo combobox1
   // neste componente TIMER o tempo foi padronizado em 2000 mili segundos
   Banco.Query.Close;
   Banco.Query.SQL.Clear;
   Banco.Query.SQL.Add('SELECT * FROM tb_computadores WHERE
cod_grupo='+ComboBox1.Text+'');
   Banco.Query.Open;
   // posiciona no primeiro registro da selecao
   Banco.Query.First;
   // executa todos os ativos de rede do grupo
   while not Banco.Query.Eof do
   begin
      IP := Banco.Query.Fields[4].AsString; // pega o conteudo do IP
      Ping1.DnsLookup(IP); // executa o componente ping
      // insere as variaveis HOST, PACOTE, RESPOSTA, DATA na tabela
TB_HISTORICO
      // caracter #34 é igual aspas simples
      Banco.Query1.Close;
      Banco.Query1.SQL.Clear;
      Banco.Query1.SQL.ADD('INSERT INTO tb_historico (Host, Pacote,
Resposta, Data, Hora)VALUES('+#34+IP+#34+', '+#34+
IntToStr(Ping1.Reply.DataSize) +#34+', '+#34+ IntToStr(Ping1.Reply.RTT)
+#34+', '+#34+DateToStr(now)+#34+', '+#34+TimeToStr(now)+#34+')');
      Banco.Query1.Open;
      Banco.Query.Next;
   end;
end;

procedure TPingTstForm.Porgrupo1Click(Sender: TObject);
begin
  //chama o formulario com a opcao de relatorio por ativo de rede
   FormRelatorioAtivo.ShowModal;
```

```
end;

procedure TPingTstForm.PageControl1Change(Sender: TObject);
begin
  // limpa o campo memo
  DisplayMemo.Clear;
end;

procedure TPingTstForm.ComboBox1DropDown(Sender: TObject);
begin
  // Traz todos os grupos cadastrados e adiciona no componente combobox
  ComboBox1.Items.Clear;
  Banco.tb_grupo.First;
  while not Banco.tb_Grupo.Eof do
  begin
     ComboBox1.Items.Add(Banco.tb_Grupo.FieldByName('Cod_grupo').AsString);
     Banco.tb_Grupo.Next;
  end;
end;

end.
```